Kementerian Pendidikan Dan Kebudayaan
Republik Indonesia
2013

PEMOGRAMAN BERORIENTASI OBJEK

Semester
2

Untuk SMK / MAK Kelas XI

Penulis : **Eko Subiyantoro**
Editor Materi : **Joko Pitono**
Editor Bahasa :
Ilustrasi Sampul :
Desain & Ilustrasi Buku : **PPPPTK BOE Malang**

Pusat Pengembangan Pemberdayaan Pendidik dan Tenaga Kependidikan Bidang Otomotif dan Elektronika:

Jl. Teluk Mandar, Arjosari Tromol Pos 5, Malang 65102, Telp. (0341) 491239, (0341) 495849, Fax. (0341) 491342, Surel: vedcmalang@vedcmalang.or.id, Laman: www.vedcmalang.com

DISKLAIMER (*DISCLAIMER*)

Penerbit tidak menjamin kebenaran dan keakuratan isi/informasi yang tertulis di dalam buku tek ini.Kebenaran dan keakuratan isi/informasi merupakan tanggung jawab dan wewenang dari penulis.

Penerbit tidak bertanggung jawab dan tidak melayani terhadap semua komentar apapun yang ada didalam buku teks ini.Setiap komentar yang tercantum untuk tujuan perbaikan isi adalah tanggung jawab dari masing-masing penulis.

Setiap kutipan yang ada di dalam buku teks akan dicantumkan sumbernya dan penerbit tidak bertanggung jawab terhadap isi dari kutipan tersebut. Kebenaran keakuratan isi kutipan tetap menjadi tanggung jawab dan hak diberikan pada penulis dan pemilik asli.Penulis bertanggung jawab penuh terhadap setiap perawatan (perbaikan) dalam menyusun informasi dan bahan dalam buku teks ini.

Penerbit tidak bertanggung jawab atas kerugian, kerusakan atau ketidaknyamanan yang disebabkan sebagai akibat dari ketidakjelasan, ketidaktepatan atau kesalahan didalam menyusun makna kalimat didalam buku teks ini.

Kewenangan Penerbit hanya sebatas memindahkan atau menerbitkan mempublikasi, mencetak, memegang dan memproses data sesuai dengan undang-undang yang berkaitan dengan perlindungan data.

Katalog Dalam Terbitan (KDT)
Rekayasa Perangkat Lunak Edisi Pertama 2013
Kementerian Pendidikan & Kebudayaan
Direktorat Jenderal Peningkatan Mutu Pendidik & Tenaga Kependidikan, th. 2013: Jakarta

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan yang Maha Esa atas tersusunnya buku teks ini, dengan harapan dapat digunakan sebagai buku teks untuk siswa Sekolah Menengah Kejuruan (SMK) Bidang Studi Keahlian Rekayasa Perangkat Lunak.

Penerapan kurikulum 2013 mengacu pada paradigma belajar kurikulum abad 21 menyebabkan terjadinya perubahan, yakni dari pengajaran (*teaching*) menjadi BELAJAR (*learning*), dari pembelajaran yang berpusat kepada guru (*teachers-centered*) menjadi pembelajaran yang berpusat kepada peserta didik (*student-centered*), dari pembelajaran pasif (*pasive learning*) ke cara belajar peserta didik aktif (*active learning-CBSA*) atau *Student Active Learning-SAL*.

Buku teks ″Pemrograman Berorientasi Obyek″ ini disusun berdasarkan tuntutan paradigma pengajaran dan pembelajaran kurikulum 2013 diselaraskan berdasarkan pendekatan model pembelajaran yang sesuai dengan kebutuhan belajar kurikulum abad 21, yaitu pendekatan model pembelajaran berbasis peningkatan keterampilan proses sains.

Penyajian buku teks untuk Mata Pelajaran ″Pemrograman Berorientasi Obyek ″ ini disusun dengan tujuan agar supaya peserta didik dapat melakukan proses pencarian pengetahuan berkenaan dengan materi pelajaran melalui berbagai aktivitas proses sains sebagaimana dilakukan oleh para ilmuwan dalam melakukan eksperimen ilmiah (penerapan scientifik), dengan demikian peserta didik diarahkan untuk menemukan sendiri berbagai fakta, membangun konsep, dan nilai-nilai baru secara mandiri.

Kementerian Pendidikan dan Kebudayaan, Direktorat Pembinaan Sekolah Menengah Kejuruan, dan Direktorat Jenderal Peningkatan Mutu Pendidik dan Tenaga Kependidikan menyampaikan terima kasih, sekaligus saran kritik demi kesempurnaan buku teks ini dan penghargaan kepada semua pihak yang telah berperan serta dalam membantu terselesaikannya buku teks siswa untuk Mata Pelajaran basis data kelas XI / Semester 1 Sekolah Menengah Kejuruan (SMK).

Jakarta, 12 Desember 2013

Menteri Pendidikan dan Kebudayaan

Prof. Dr. Mohammad Nuh, DEA

DAFTAR ISI

GLOSARIUM

Abstract class adalah class yang mempunyai sedikitnya satu abstract method. Abstract class hanya bisa digunakan sebagai super class, dan dapat diturunkan dari class abstract lainnya.

Abstract method adalah method yang belum mempunyai implementasi.

Array adalah suatu kumpulan data pada suatu variabel. Array digunakan untuk membuat variabel bisa menampung beberapa data dengan tipe data yang sama alias satu tipe data.

Ascending adalah mengurutkan data dari kecil ke besar

Catch digunakan untuk menangkap kesalahan atau bug yang terjadi dalam block try.

Class merupakan suatu blueprint atau cetakan untuk menciptakan suatu instant dari object.

Class file merupakan representasi dari file dan direktori (path).

Class FileDescriptor digunakan untuk menunjukkan descriptor dari file yang aktif.

Class Math berisi method untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma.

Class Proccess menyediakan metode untuk melakukan input dari proses, melakukan output ke proses, menunggu proses untuk menyelesaikan, memeriksa status keluar dari proses, dan menghancurkan (membunuh) proses.

Class StringBuffer adalah pasangan class String yang menyediakan banyak fungsi string yang umum.

Class System menyediakan beberapa field dan method bermanfaat, seperti standard input, standard output dan sebuah method yang berguna untuk mempercepat penyalinan bagian sebuah array.

Class Wrapper adalah representasi objek sederhana dari variabel- variable non-objek yang sederhana. Ada 10 tipe data Wrapper, yaitu Boolean, Byte, Character, Double, Float, Integer, Long, Number, Short, dan Void.

Collection merupakan istilah umum yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan suatu teknik tertentu pula.

Descending adalah mengurutkan data dari besar ke kecil

Exception adalah sebuah event yang menjalankan alur proses normal pada program.

Extends Class adalah class yang mewarisi sifat dari sifat-sifat yang dimiliki oleh superclass.

Finally merupakan keyword pada class exception handling yang menunjukan bahwa blockprogram tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada.

Himpunan (set) adalah kumpulan Object yang mana tidak boleh ada dua dari objek yang sama di dalam satu himpunan.

Inheritance adalah pewarisan method dan atribut dari super-class kepada sub-classnya

Interface merupakan sekumpulan dari method-method yang dibuat, namun belum ada operasi di dalam tubuh method tersebut.

IOException berfungsi menginformasikan pada compiler ada proses operasi input yang mungkin failed.

Java String merupakan salah satu kelas dasar yang disediakan oleh Java untuk memanipulasi karakter.

Keyword Extends digunakan untuk melakukan proses penurunan terhadap suatu class.

Konstruktor merupakan method khusus yang dipakai oleh Java untuk membuat sebuah object didalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor.

List merupakan pengelompokan berdasarkan urutan seperti layaknya array, karena itu ia memiliki posisi awal dan juga posisi akhir.

Map merupakan Object yang memetakan object ke nilai.

Method adalah bagian-bagian kode yang dapat dipanggil ole program utama atau dari method lainnya untuk menjalankan fungsi yang spesifik.

Modifier digunakan untuk menentukan sifat dari suatu kelas dan menentukan preveledge (hak akses) dari kelas lain.

Multiple inheritance adalah pewarisan dimana kelas yang diturunkan lebih dari satu kelas yang berbeda (super-class lebih dari satu).

Overloading adalah mendefinisikan beberapa metode yang memiliki nama sama tetapi dengan sidik yang berbeda.

Overriding adalah menyediakan suatu implementasi baru untuk suatu metode didalam subkelas.

PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusial.

Queue merupakan model pengelompokan berdasarkan metode antrian suatu prioritas tertentu(contoh FIFO-First In First Out).

Quick Sort adalah algoritma yang dijalankan sebagai akibat dari terlalu banyaknyadaftar yang diurutkan, dengan menghasilkan lebih banyak daftar yang diurutkan sebagai output.

Selection Sort merupakan Kombinasi antara sorting dan searching.

Set merupakan pengelompokan mengikuti model himpunan dimana setiap anggota-nya harus unik.

Single inheritance yaitu pewarisan yang jumlah kelas dasarnya (super-class) hanya satu, tetapi kelas turunannya bisa lebih dari satu.

SortedMap adalah sebuap Map yang memelihara elemen key-nya terurut secara ascending.

SortedSet adalah Sebuat set yang memelihara pemetaan elemennya secara ascennding.

Sorting adalah proses menyusun elemen – elemen dengan tata urut tertentu dan proses tersebut terimplementasi dalam bermacam aplikasi.

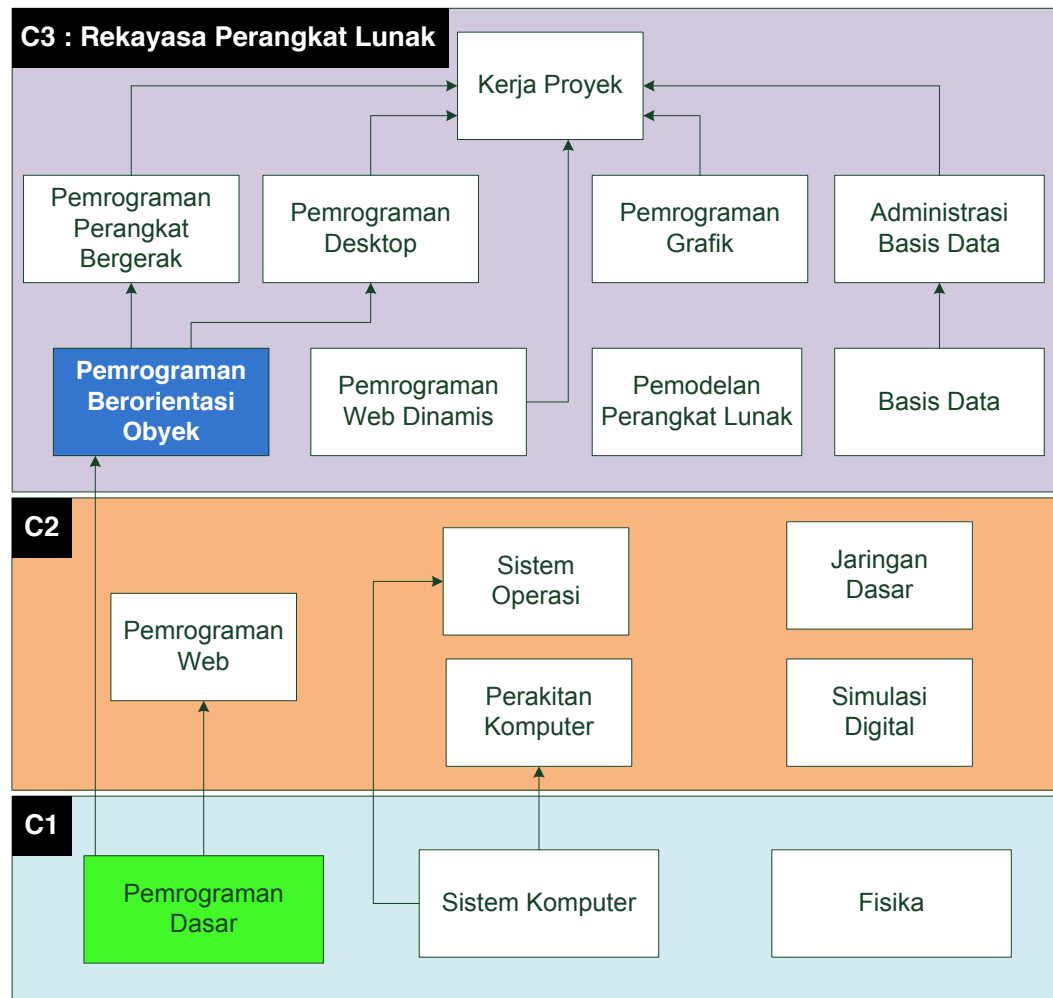
StringBuffer () digunakan untuk mengkonstruksi buffer string kosong

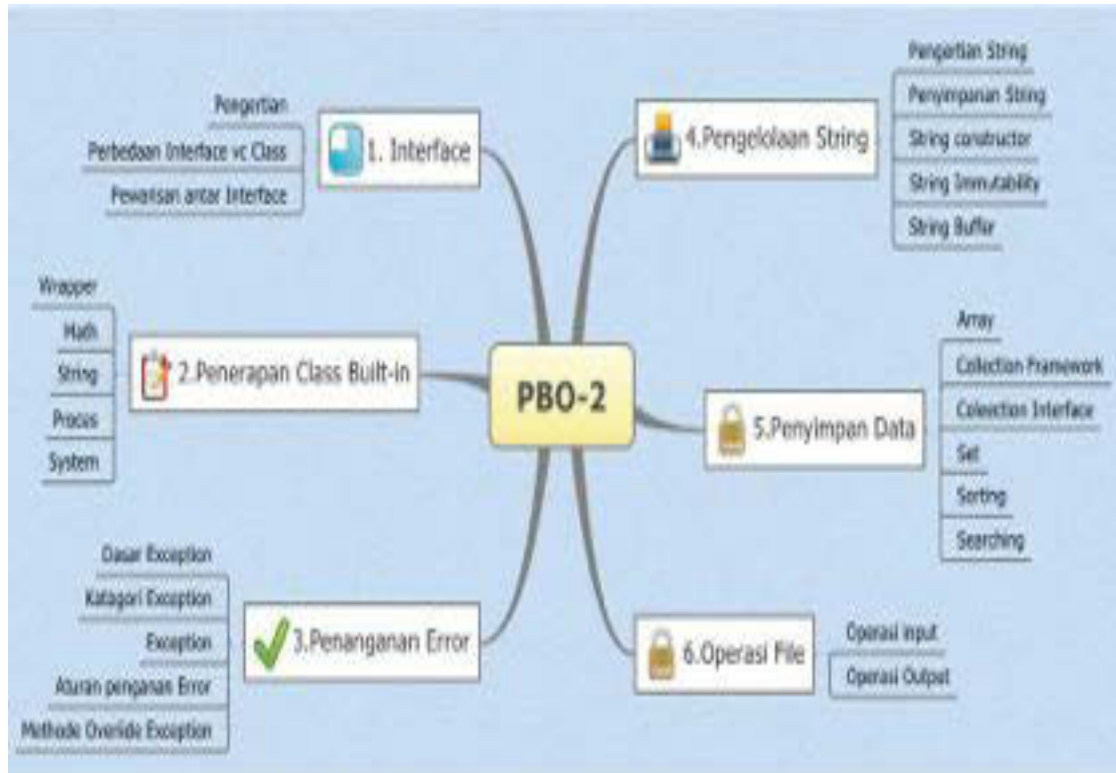
StringBuffer (int leght) digunakan untuk mengkonstruksi buffer string kosong (tanpa karakter) dan kapasitas ditentukan oleh parameter length.

Throw digunakan untuk melemparkan suatu bug yang dibuat secara manual.

Try digunakan dalam suatu blockprogram. Keyword ini digunakan untuk mencoba menjalankan blockprogram, kemudian mengenai dimana munculnya kesalahan yang ingin diproses.

Throws digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya.

PETA KEDUDUKAN BUKU

Keterangan	
C1	Kelompok mata pelajaran Dasar Bidang Keahlian Teknologi Informasi dan Komunikasi
C2	Kelompok mata pelajaran Dasar Program Keahlian Teknik Komputer dan Informatika
C3	Kelompok mata pelajaran Paket Keahlian Rekayasa Perangkat Lunak
(biru)	Mata pelajaran Pemrograman Berorientasi Obyek Semester 2
(hijau)	Mata pelajaran prasyarat

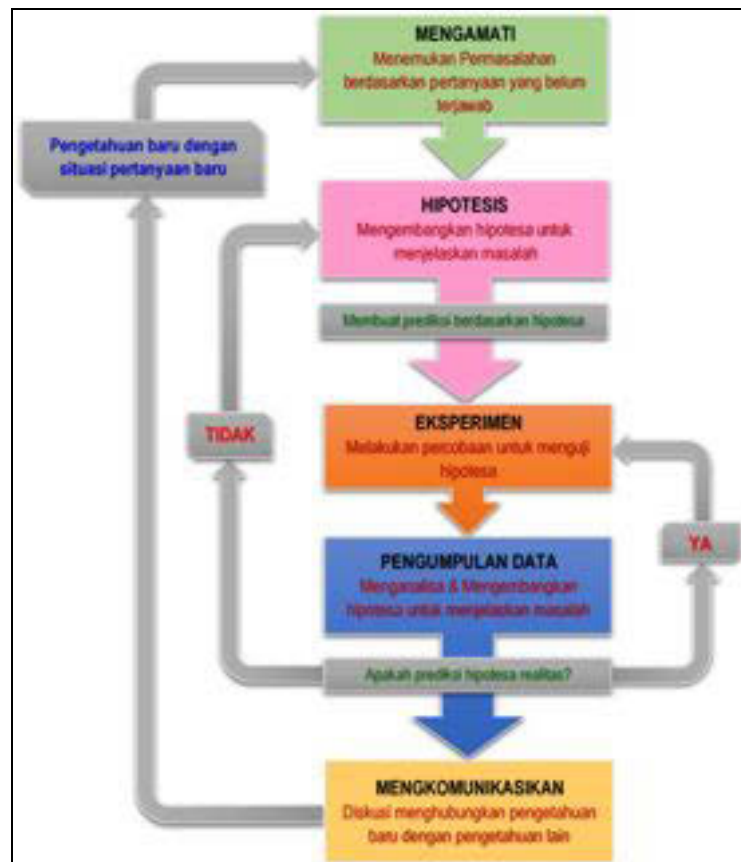
Peta Konsep : Pemrograman Berorientasi Obyek Kelas XI Semester 2

Keterangan	
KD 3.8- 4.8	Interface
KD 3.9- 4.9	Penerapan Class Built-in
KD 3.10- 4.10	Penanganan Error
KD 3.11- 4.11	Pengelolaan String
KD 3.12- 4.12	Penyimpan Data
KD 3.13- 4.13	Operasi File

BAB I PENDAHULUAN

A. Deskripsi.

Pemrograman berorientasi objek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Ini adalah jenis pemrograman di mana programmer mendefinisikan tidak hanya tipe data dari sebuah struktur data, tetapi juga jenis operasi (fungsi) yang dapat diterapkan pada struktur data. Dengan cara ini, struktur data menjadi objek yang meliputi data dan fungsi. Selain itu, pemrogram dapat membuat hubungan antara satu benda dan lainnya. Sebagai contoh, objek dapat mewarisi karakteristik dari objek lain.

Salah satu keuntungan utama dari teknik pemrograman berorientasi obyek atas teknik pemrograman prosedural adalah bahwa memungkinkan programmer untuk membuat modul yang tidak perlu diubah ketika sebuah jenis baru objek ditambahkan.Seorang pemrogram hanya dapat membuat objek baru yang mewarisi banyak fitur dari objek yang sudah ada.Hal ini membuat program object-oriented lebih mudah untuk memodifikasi.

Pembelajaran pemrograman berorientasi obyek ini menggunakan metode pendekatan saintifik.Dalam pendekatan ini praktikum atau *e*ksperimen berbasis sains merupakan bidang pendekatan ilmiah dengan tujuan dan aturan khusus, dimana tujuan utamanya adalah untuk memberikan bekal ketrampilan yang kuat dengan disertai landasan teori yang realistis mengenai fenomena yang akan kita amati. Ketika suatu permasalahan yang hendak diamati memunculkan pertanyaan-pertanyaan yang tidak bisa terjawab, maka metode eksperimen ilmiah hendaknya dapat memberikan jawaban melalui proses yang logis. Proses-proses dalam pendekatan scientifik meliputi beberapa tahapan yaitu: mengamati, hipotesis atau menanya, mengasosiasikan atau eksperimen, mengumpulkan atau analisa data dan mengkomunikasikan. Proses belajar pendekatan eksperimen pada hakekatnya merupakan proses berfikir ilmiah untuk membuktikan hipotesis dengan logika berfikir.

Gambar 1. Diagram Proses Metode Saintifik-Eksperimen Ilmiah

B. Prasyarat.

Untuk kelancaran pencapaian kompetensi dalam mata pelajaran pemrograman berorientasi obyek ini dibutuhkan beberapa peryaratan baik pengetahuan maupun ketrampilan dasar. Persyaratan tersebut antara lain ialah: Peserta didik telah menguasai mata pelajaran pemrograman dasar. Konsep dan algoritma pemrograman ini dibutuhkan untuk mendukung implementasi pemrograman berorientasi obyek.Disamping itu peserta didik mempunyai kompetensi dalam hal pemanfaatan teknologi informasi, seperti mengoperasikan hardware komputer dan mengoperasikan perangkat lunak aplikasi. Perangkat lunak aplikasi tersebut antar lain ialah pengolah data untuk menganalisis data hasil eksperimen, pengolah kata untuk membuat laporan dan aplikasi presentasi untuk mengkomunikasikan dan mempresentasikan hasil laporan.

C. Petunjuk Penggunaan.

Buku pedoman siswa ini disusun berdasarkan kurikulum 2013 yang mempunyai ciri khas penggunaan metode scientifik. Buku ini terdiri dari dua bab yaitu bab satu pendahuluan dan bab dua pembelajaran. Dalam bab pendahuluan beberapa yang harus dipelajari peserta didik adalah diskripsi mata pelajaran yang berisi informasi umum, rasionalisasi dan penggunaan metode scientifik. Selanjutnya pengetahuan tentang persyaratan, tujuan yang diharapkan, kompetensi inti dan dasar yang akan dicapai serta test kemampuan awal.

Bab dua menuntun peserta didik untuk memahami deskripsi umum tentang topik yang akan dipelajari dan rincian kegiatan belajar sesuai dengan kompetensi dan tujuan yang akan dicapai. Setiap kegiatan belajar terdiri dari tujuan dan uraian materi topik pembelajaran, tugas serta test formatif. Uraian pembelajaran berisi tentang diskripsi pemahaman topik materi untuk memenuhi kompetensi pengetahuan.Uraian pembelajaran juga menjelaskan deskripsi unjuk kerja atau langkah-langkah logis untuk memenuhi kompetensi skill.

Tugas yang harus dikerjakan oleh peserta didik dapat berupa tugas praktek, eksperimen atau pendalaman materi pembelajaran. Setiap tugas yang dilakukan melalui beberapa tahapan saintifik yaitu : 1) melakukan pengamatan setiap tahapan unjuk kerja 2) melakukan praktek sesuai dengan unjuk kerja 3) mengumpulkan data yang dihasilkan setiap tahapan 4) menganalisa hasil data menggunakan analisa diskriptif 5) mengasosiasikan beberapa pengetahuan dalam uraian materi pembelajaran untuk membentuk suatu kesimpulan 6) mengkomunikasikan hasil dengan membuat laporan portofolio. Laporan tersebut merupakan tagihan yang akan dijadikan sebagai salah satu referensi penilaaian.

D. Tujuan Akhir.

Setelah mempelajari uraian materi dalam bab pembelajaran dan kegiatan belajar diharapkan peserta didik dapatmemiliki kompetensi sikap, pengetahuan dan ketrampilan yang berkaitan dengan materi:

- ✓ Interface
- ✓ Penerapan class built-in
- ✓ Penanganan Error
- ✓ Pengelolaan String
- ✓ Penyimpanan Data

E. Kompetensi Inti Dan Kompetensi Dasar

1. **Kompetensi Inti 1 :** Menghayati dan mengamalkan ajaran agama yang dianutnya.

   Kompetensi Dasar :

   1.1. Memahami nilai-nilai keimanan dengan menyadari hubungan keteraturan dan kompleksitas alam dan jagad raya terhadap kebesaran Tuhan yang menciptakannya

   1.2. Mendeskripsikan kebesaran Tuhan yang menciptakan berbagai sumber energi di alam

   1.3. Mengamalkan nilai-nilai keimanan sesuai dengan ajaran agama dalam kehidupan sehari-hari.

2. **Kompetensi Inti 2:** Menghayati dan Mengamalkan perilaku jujur, disiplin, tanggung jawab, peduli (gotong royong, kerjasama, toleran, damai), santun, responsif dan proaktif dan menunjukan sikap sebagai bagian dari solusi atas berbagai permasalahan dalam berinteraksi secara efektif dengan lingkungan sosial dan alam serta dalam menempatkan diri sebagai cerminan bangsa dalam menempatkan diri sebagai cerminan bangsa dalam pergaulan dunia.

   Kompetensi Dasar:

   2.1. Menunjukkan perilaku ilmiah (memiliki rasa ingin tahu; objektif; jujur; teliti; cermat; tekun; hati-hati; bertanggung jawab; terbuka; kritis; kreatif; inovatif dan peduli lingkungan) dalam aktivitas sehari-hari sebagai wujud implementasi sikap dalam melakukan percobaan dan berdiskusi

   2.2. Menghargai kerja individu dan kelompok dalam aktivitas sehari-hari sebagai wujud implementasi melaksanakan percobaan dan melaporkan hasil percobaan.

3. **Kompetensi Inti 3:** Memahami, menerapkan dan menganalisis pengetahuan faktual, konseptual dan prosedural berdasarkan rasa ingin tahunya tentang ilmu pengetahuan, teknologi, seni, budaya, dan humaniora dalam wawasan kemanusiaan, kebangsaan, kenegaraan, dan peradaban terkait penyebab fenomena dan kejadian dalam bidang kerja yang spesifik untuk memecahkan masalah.

Kompetensi Dasar:

3.8. Memahami pembuatan interface

3.9. Menganalisis pemanfaatan class built-in

3.10. Memahami mekanisme penanganan kesalahan

3.11. Memahami string dan berbagai propertinya

3.12. Memahami data collection sebagai media penyimpanan data.

3.13. Menerapkan operasi file dan Input Output(IO)

4. **Kompetensi Inti 4:** Mengolah, menalar, dan menyaji dalam ranah konkret dan ranah abstrak terkait dengan pengembangan dari yang dipelajarinya di sekolah secara mandiri, dan mampu melaksanakan tugas spesifik dibawah pengawasan langsung.

   Kompetensi Dasar:

   4.8 Menyajikan hasil pembuatan aplikasi dengan interface

   4.9 Menyajikan beberapa class built-in dan penerapannya dalam memecahkan masalah

   4.10 Mengolah penanganan error dalam mendeteksi kesalahan program

   4.11 Mengolah data String dan berbagai propertinya

   4.12 Menyajikan data colection sebagai penyimpan data

   4.13 Menyajikan operasi file dan operasi Input Output

F. Cek Kemampuan Awal

1. Jelaskan perbedaan perbedaan interface dengan class !
2. Jelaskan fungsi dan manfaat class-class built-in!
3. Jelaskan secara singkat cara penanganan error!
4. Jelaskan cara pengelolaan data String !
5. Jelaskan secara singkat konsep konsep data collection sebagai penyimpan data !
6. Jelaskan secara singkat operasi file dalam class !

BAB II KEGIATAN BELAJAR

1. Kegiatan Belajar 1 : Interface (Pengertian)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1 ini siswa diharapkan dapat :

1). Memahami pengertian interface

2). Menyajikan hasil pembuatan interface sederhana

B. Uraian Materi

1) Pengantar Interface

Di kelas kita sudah belajar apa interface tersebut, untuk mengulang materi di kelas, semoga postingan saya tentang interface kali ini bisa memperjelas tentang konsep interface yang kita pelajari di kelas praktikum dan penjelasan tugas yang sudah dikumpulkan minggu lalu.

Kenapa kita butuh interface? Sebagai pengantar kita harus mengetahui apa yang disebut interface dan kegunaannya dalam pemrograman java khususnya pemrograman berorientasi objek, karena kita akan bermain banyak dengan objek tersebut.

Interface adalah jenis khusus dari blok yang hanya berisi method signature atau constant. Interface mendefinisikan sebuah signature dari sebuah kumpulan method tanpa tubuh. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-class tanpa memperhatikan lokasinya dalam hirarki class untuk mengimplementasikan sifat-sifat yang umum. Dengan catatan bahwa interface juga menunjukkan polimorfisme, dikarenakan program dapat memanggil method interface dan versi yang tepat dari method yang akan dieksekusi tergantung dari tipe object yang melewati pemanggil method interface.

Untuk lebih mudah memahami, interface merupakan sekumpulan dari method-method yang dibuat, namun belum ada operasi di dalam tubuh method

tersebut.Interface bisa diturunkan atau diwariskan kepada class yang ingin memakai method yang ada dalam masing-masing interface tersebut dengan keyword **extends** [interface yang didefinisikan]. Sebuah class dapat mengimplementasikan 1 interface yang sudah dibuat dengan keyword **implements.**

✓ **Ciri-ciri Interface**

Ciri-ciri dari interface adalah sebagai berikut :

- Method interface tidak punya tubuh, sebuah interface hanya dapat mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class lainnya, mereka didefinisikan secara independent.
- Tidak bisa membuat instance atau objek baru dari sebuah interface.
- Ciri umum lain adalah baik interface maupun class dapat mendefinisikan method. Bagaimanapun, sebuah interface tidak memiliki kode implementasi sedangkan class memiliki salah satunya.

✓ **Pendeklarasian Interface**

Contoh pendeklarasian interface adalah sebagai berikut :

Listing Program

```
1 public interface InterfaceA{
2 String atributA = "Ini konstanta dari interface A";
3 void methodSatuA();
4 String methodSatuA();
5 }
```

✓ **Implementasi Interface**

Cara menggunakan interface pada kelas lain, harus menggunakan keyword **implements.** Deklarasi implements interface sebagai berikut :

1. Dalam project yang telah dibuat sebelumnya, buatlah satu package baru dengan nama (Misal : **tugas01**).
2. Dalam package tersebut, buatlah interface dengan nama InterfaceA.

Listing Program

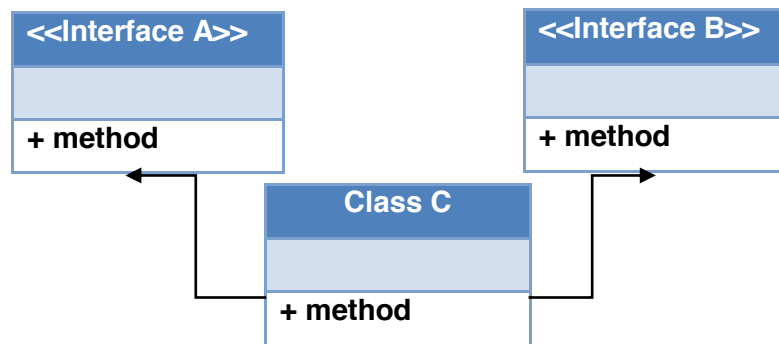
```
1 public interface InterfaceA{
2 String atributA = "Ini konstanta dari interface A";
3 void methodSatuA();
4 String methoDuaA();
5 }
```

3. Selanjutnya buat class baru dengan nama CobaPertama yang mengimplementasikan sifat dari InterfaceA.

Listing Program

```
1 public class CobaPertama implements InterfaceA{
2 public void methodSatuA(){
3 System.out.println("Isi method pertama dari interface A");
4 }
5  public String methodDuaA(){
6  return atributA;
7  }
8  Public static void main (String []args){
9  CobaPertama obj = new CobaPertama();
10 obj.methodSatuA();
11 String pesan = obj.methodDuaA();
12 System.out.println(pesan);
13 }
14 }
```

2) Multiple Interface

Java tidak memperkenankan adanya multiple inheritance, tetapi java memperbolehkan multiple interface.Dibawah ini adalah ilustrasi multiple interface.

✓ **Implementasi Multiple Interface**

Keuntungan menggunakan interface (implements) dibandingkan dengan menggunakan konsep pewarisan adalah jika dalam pewarisan(extend) hanya mengenal single inheritance maka dalam interface kita dapat menggunakan konsep multiple interface. Jadi suatu class dapat mengimplemen lebih dari satu interface.

Contoh penggunaan multiple adalah sebagai berikut :

1. Pada package (**tugas01**) yang sudah kita kerjakan sebelumnya, tambahkan class interface dengan nama InterfaceB.

Listing Program

```
1 public interface InterfaceB{
2 void methodSatuB();
3 void methodDuaB();
4 }
```

2. Buat class baru dengan nama CobaKedua.

Listing Program

```
1 public class CobaKedua implements InterfaceA{
2 public void methodSatuA(){
3  System.out.println("Isi method pertama dari interface
   A");
4 }
5  public String methodDuaA(){
6  return atributA;
7  }
8  Public void methodSatuB(){
```

```
9  System.out.println("Isi method pertama dari Interface
   B");
10 }
11 Public void methodDuaB(){
12 System.out.println("Isi  method  kedua  dari  Interface
   B");
13 }
14 Public static void main (String []args){
15 CobaKedua obj = new CobaKedua();
16 String pesan = obj.methodDuaA():
17 System.out.println(pesan);
18 obj.methodSatuA();
19 obj.methodSatuB();
20 obj.methodDuaB();
21 }
22 }
```

C. Rangkuman

Interface merupakan kumpulan dari method-method yang belum terdapat operasi di dalam tubuh method tersebut.Interface bisa diturunkan atau diwariskan kepada class yang ingin memakai method yang ada dalam masing-masing interface tersebut dengan keyword **extends** [interface yang didefinisikan]. Sebuah class dapat mengimplementasikan 1 interface yang sudah dibuat dengan keyword **implement.**Interface dapat mendefinisikan konstanta. Interface juga tidak dapat membuat instance atau objek baru dari sebuah interface.Baik interface maupun class dapat mendefinisikan method. dalam java terdapat multiple interface, dimana dalam satu class dapat mengimplementasikan lebih dari satu interface.

D. Tugas

Tugas 1

Buatlah program berikut :

- Buat class Operasi, Kalkulator dan UjiKalkulator. Dimana class Operasi adalah class interface yang terdapat method sebagai berikut:
  Public void penjumlahan ();
  Public void pengurangan ();
- Class Kalkulator digunakan untuk implements class Operasi
- Class UjiKalkulator digunakan untuk menguji class Kalkulator

Tugas 2

Buatlah program berikut :

Buat class Superhero, dimana class Superhero adalah class interface yang terdapat method sebagai berikut:

- Public void superman ();
- Public void spiderman ();
- Public void Thor ();

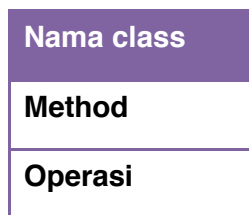
Tugas 3

Buatlah program berikut :

Buat class Bank, dimana class Bank adalah class interface yang terdapat method sebagai berikut:

- Public void menabung ();
- Public void Transfer ();
- Public void Menarik ();

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   A. Interface
   B. Extend
   C. Implements
2. Sebutkan ciri-ciri dari Interface !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Interface

....

....

....

....

b) Extend

…..

....

....

....

c) Implemets

....

....

....

2. Kegiatan Belajar 2 : Interface (Perbedaan Interface dan Class)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 2 ini siswa diharapkan dapat :

1). Memahami perbedaan Interface dan Class

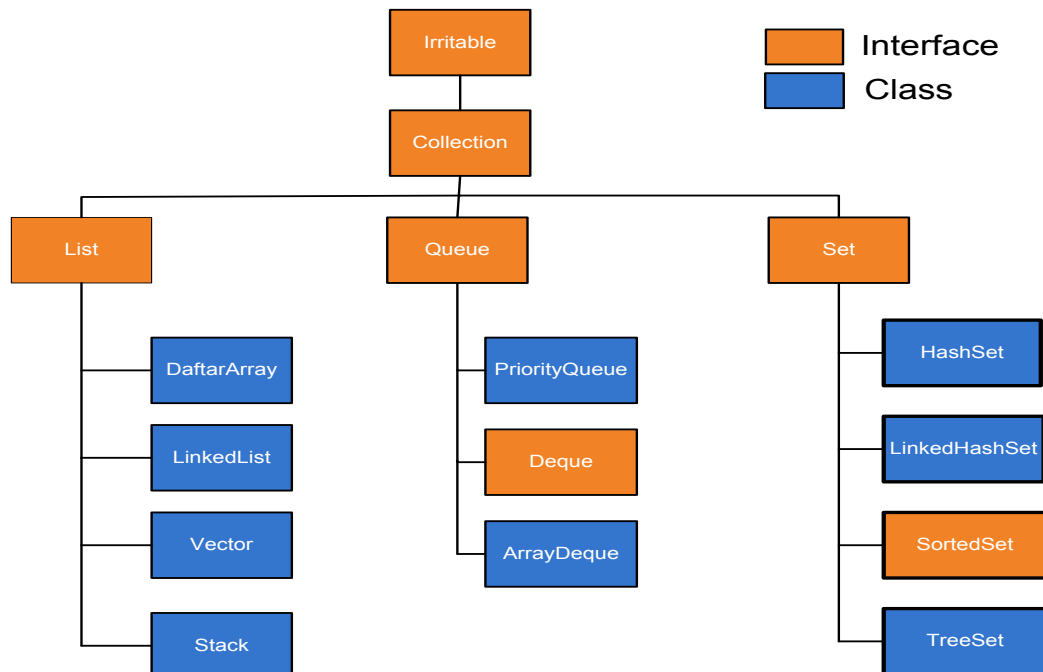
2). Menyajikan hasil pembuatan Interface dan abstract class

B. Uraian Materi

1) Class

Class merupakan suatu blueprint atau cetakan untuk menciptakan suatu instant dari object. Class juga merupakan grup suatu object dengan kemiripan attribute, properties, behavior, dan relasi ke object lain.

Contoh: Class Hewan, Class Manusia, Class Buah.

2) Perbedaan Interface dan Class

3) Abstract Class

Abstract class adalah class yang mempunyai sedikitnya satu abstract method. Abstract class hanya bisa digunakan sebagai super class, dan dapat diturunkan dari class abstract lainnya. Untuk mendeklarasikan sebuah abstract class digunakan keyword abstract, **[abstract] class [class_name].**

Sebuah abstract class pada dasarnya tidak jauh berbeda dengan class lainnya, yakni juga berisi method yang menggambarkan karakteristik dari kelas abstract tersebut.Perbedaannya yaitu sebuah abstract class bisa berisi method tanpa diimplementasikan, artinya sebuah method tanpa body. Method seperti ini disebut method abstract.

4) Implementasi Abstract Class

Abstract class tidak bisa dibuat objectnya atau tidak dapat di instasiasi. Object hanya bisa dibuat dari non-abstract class (concrete class). Konsekuensinya suatu abstract class harus diturunkan dimana pada subclass tersebut berisi implementasi dari abstract method yang ada di superclass.

Sintaks dalam membuat abstract class adalah sebagai berikut :

Listing Program

```
1 public abstract class Hewan
2 {
3 …//definisi class
4 }
```

Sintaks dalam membuat method abstract class adalah sebagai berikut :

Listing Program

```
1 public abstract class Hewan{
2 void Bernafas(){
3 System.out.println(“Bernafas”);
4 }
5 }
```

5) Abstract Method

Abstract method adalah method yang belum mempunyai implementasi.Kita dapat menyatakan suatu method abstract dengan membutuhkan keyword abstract pada deklarasi method tersebut.

Secara umum sintaks pendeklarasian abstract method adalah sebagai berikut :

Listing Program

```
1 abstract class Seniman{
2 public abstract void berkesenian();
3 public void tidur(){
4 System.out.println("Zzz…");
5 }
6 }
```

Listing Program

```
1 class Penyanyi extends Seniman{
2 public void berkesenian(){
3 System.out.println("Tralala-trilili…");
4 }
5 }
```

Listing Program

```
1 public class Explain{
2 public static void main(String args []){
3 Penyanyi Joshua = new Penyanyi();
4 Joshua.berkesenian();
5 }
6 }
```

6) Perbedaan Abstract Class dan Interface

Abstract Class	Interface
1. Bisa berisi abstract dan non-abstract method.	1. Hanya boleh berisi abstract method.
2. Kita harus menuliskan sendiri modifiernya.	2. Kita tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah *public* dan *abstract*.
3. Dapat mendeklarasikan constant dan instance variable.	3. Hanya bisa mendeklarasikan constant. Secara implisit variable yang dideklarasikan di interface bersifat public, static dan final.
4. Method boleh bersifat static.	4. Method tidak boleh bersifat static.
5. Method boleh bersifat final.	5. Method tidak boleh bersifat final.
6. Suatu abstract class hanya bisa meng-*extend* satu abstract class lainnya.	6. Suatu interface bisa meng-*extend* satu atau lebih interface lainnya.
7. Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-implement beberapa interface.	7. Suatu interface hanya bisa meng-*extend* interface lainnya. Dan tidak bisa meng-*implement* class atau interface lainnya.

C. Rangkuman

Abstract class adalah class yang mempunyai sedikitnya satu abstract method. Abstract class hanya digunakan sebagai super class dan dapat diturunkan dari class abstract lainnya. Pendeklarasian abstract class dengan menggunakan keyword **abstract class.** Abstract class berisi method tanpa diimplementasikan yang disebut dengan method abstract.

Perbedaan utama dari sebuah interface dan abstract class adalah method interface tidak mempunyai tubuh, interface hanya mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class lainnya, melainkan secara secara independen.

D. Tugas

Tugas 1

Buatlah program berikut :

'Class Bentuk' akan digunakan sebagai abstract class untuk class turunannya yaitu 'Class Lingkaran', 'Class Segitiga', dan 'Class SegiEmpat'

Tugas 2

Buatlah program berikut :

'Class Kue' akan digunakan sebagai abstract class untuk class turunannya yaitu 'Class Bolu', 'Class Tart', dan 'Class Brownies'

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   a. Class
   b. Abstract class
   c. Abstract method
2. Jelaskan perbedaan interface dan abstract class yang anda ketahui!

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Class

....

....

....

....

b) Abstract Class

....

....

....

....

c) Abstract method

....

....

....

....

d) Method Abstract

....

....

....

LJ- 02 : Perbedaan interface dan abstract class :

....

....

....

....

....

....

....

G. Lembar Kerja Siswa

3. Kegiatan Belajar 3 : Interface (Pewarisan Antar Interface)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1). Memahami pewarisan antar interface

2). Menyajikan hasil pembuatan pewarisan antar interface

B. Uraian Materi

1) Pewarisan antar Interface

Dalam OOP sering kali kita mendengar istilah pewarisan (Inheritance), yaitu sebuah sub-class akan mewarisi behavior(method) ataupun attribut yang ada di dalam super-class nya .

Penggunaan inheritance dapat dilakukan secara overriding ataupun secara overloading method. Overloading berarti mendefinisikan beberapa metode yang memiliki nama sama tetapi dengan sidik yang berbeda. Sedangkan overriding berarti menyediakan suatu implementasi baru untuk suatu metode didalam subkelas.Ini menunjukan bahwa secara konsep super-class hanya menyediakan method-method yang belum terdefinisi secara explicit (jelas), sehingga sub-class dapat memanfaatkan method-method super-class tersebut sesuai kebutuhan di setiap sub-class .

Secara umum fungsi pewarisan dikatakan sebagai metode *reuseability* :

a) Behavior(method) dideklarasikan dalam superclass, behavior tersebut otomatis diwariskan ke seluruh subclass.

b) Kita dapat menggunakan kelas yang kita buat sebelumnya (superclass) dengan membuat kelas-kelas baru (subclass) berdasar super class tersebut, dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki super class.

c) Kita dapat membuat super class yang hanya mendefinisikan behaviour namun tidak memberi implementasi dari metode-metode yang ada (framework class) , superclass seperti ini disebut kelas abstrak (dengan modifier kelas dan method abstract) dan sub-classnya disebut kelas kongkret , sehingga sub-class dapat mengimplementasi method dari

superclass sesuai dengan kebutuhan di sub-classnya tanpa mempengaruhi super-classnya.

d) Kita dapat mendefinisi method hanya sekali dan method tersebut dapat digunakan oleh seluruh subclass.

e) Sebuah subclass hanya perlu mengimplementasikan perbedaan antara dirinya sendiri dan parent-nya (super-classnya).

2) Contoh pewarisan

Listing Program

```
1  // yang dijadikan super-class
2  public abstract  class Burung{
3 // method abstract
4  public abstract void suara();
5 // method non-abstract yang akan dioverride
6  public void bisaTerbang(){
7  System.out.println("bisaTerbang donk!!");
8  }
9  }
```

Listing Program

```
1  // interface Pernafasan
2  interface Pernafasan {
3  // method yang akan di implementasikan
4  void bernafasLewat();
5  }
```

Listing Program

```
1 //sub-class pertama
2 public class bebek extends Burung implements Pernafasan{
3 //meng-override method bisaTerbang() dari kelas Burung
4 @Override
5 public void bisaTerbang() {
6 System.out.println("ups , cuma berjalan !");
7 }
8 // meng-implements method suara() dari kelas burung
9 public void suara() {
10 System.out.println("kowek-kowek");
11 }
12 //mengimplements method bernafasLewat dari
interfacePernafasan
13 public void bernafasLewat() {
14 System.out.println("lewat paru-paru");
15 }
16 }
```

Listing Program

```
1  //sub-class kedua
2  public class perkutut extends burung implements
   Pernafasan{
3  // meng-implements method suara() dari kelas burung
4  public void suara() {
5  System.out.println("kuruk-kuruk");
6  }
7 //mengimplements method bernafasLewat dari interface
   Pernafasan
8  public void bernafasLewat() {
9  System.out.println("bernafas Lewat paru-paru juga");
10 }
11 }
```

Pada pemrograman Java, sebuah inheritansi yang ditangani dengan metode abstraksi ini disebut single inheritance. Apabila kita ingin membuat pewarisan banyak (multiple inheritance) kita dapat menggunakan metode antar-muka (interface).

Dalam kode diatas dapat ditemukan sebuah interface 'Pernafasan' . Perlu diingat bahwa interface bukan sebuah kelas, untuk membuat interface kita menggunakan modifier kelas Interface.

C. Rangkuman

Dalam interface terdapat istilah pewarisan (inheritance) yaitu pewarisan method dan atribut dari super-class kepada sub-classnya. Penggunaan inheritance dapat dilakukan secara overriding dan overloading method. Untuk menggunakan inheritance pada interface,maka digunakan keyword **extends**.

Single inheritance yaitu pewarisan yang jumlah kelas dasarnya (super-class) hanya satu, tetapi kelas turunannya bisa lebih dari satu. Sedangkan multiple inheritance adalah pewarisan dimana kelas yang diturunkan lebih dari satu kelas yang berbeda (super-class lebih dari satu).

D. Tugas

Tugas 1

Buatlah program berikut :

1. Buat 3 interface dengan nama Bola, BolaTangan, dan BolaSepak.
2. Interface BolaTangan mempunyai method tangkap dan lempar.
3. Interface BolaSepak mempunyai method tangkap dan tendang.
4. Interface Bola menjadi interface yang mewarisi sifat BolaTangan dan BolaSepak.
5. Buat class Permainan yang mengimplemen interface Bola.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   a. Inheritance
   b. Single inheritance
   c. Multiple inheritance
2. Jelaskan perbedaan overloading method dan overriding method!
3. Identifikasilah kesalahan yang terdapat pada program berikut !

Listing Program

```
1 ArrayList list new ArrayList();
2 list.add ("Mataram");
3 list.add ("Yogyakarta");
4 list.add (new java.util.Date());
5 String kota list.get(0);
6 list.get (3, "Siantar");
7 System.out.println(list.get(3));
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Inheritance

....

....

....

....

b) Single inheritance

....

....

....

....

c) Multiple inheritance

....

....

....

....

LJ- 02 : Perbedaan overloading method dan overriding method :

....

....

....

....

....

....

....

LJ- 03 : Kesalahan pada program :

....

....

....

....

....

G. Lembar Kerja Siswa

4. Kegiatan 4 :Class Built-in (Math)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 4 ini siswa diharapkan dapat :

1) Menganalisis pemanfaatan class built in (class math).
2) Menyajikan beberapa class – class math dan penerapannya dalam memecahkan masalah.

B. Uraian Materi

1) Class Math

Java juga menyediakan konstanta dan method untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma. Selama method-method ini semua static, kita dapat menggunakannya tanpa memerlukan sebuah objek Math. Untuk melengkapi daftar konstanta dan method-method ini, lihatlah acuan pada dokumentasi Java API. Dibawah ini beberapa method-method umum yang sering digunakan.

2) Public Static double abs (double a)

Menghasilkan nilai mutlak. Sebuah method yang dioverload dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembaliannya juga menggunakan float atau integer atau long.

3) Public Static double random()

Method ini digunakan untuk membangkitkan suatu nilai double acak dengan rentang lebih besar atau sama dengan nol (0) dan lebih rendah dari 1 **(0 <=Math.random() < 1.0)**. Method ini digunakan untuk menuliskan suatu ekspresi sederhana untuk membangkitkan angka-angka acak dengan sembarang rentang.

4) **Public Static double max (double a, double b)**

Menghasilkan nilai maksimum, diantara dua nilai double, a dan b. Sebuah method yang di-overload dapat juga menggunakan nilai float atau integer atau juga long integer sebagai parameter, dengan kondisi tipe kembalinya juga menggunakan float atau integer atau longinteger, secara berturut-turut.

5) **Public Static double min (double a, double b)**

Menghasilkan nilai minimum diantara dua nilai double ,a dan b. Sebuah method yang di-overload dapat juga menggunakan nilai float atau integer atau juga longinteger sebagai parameter, dengan kondisi tipe kembaliannya juga menggunakan float atau integer atau longinteger, secara berturut-turut.

✓ **Public Static double ceil (double a)**

Menghasilkan bilangan bulat terkecil yang lebih besar atau sama dengan a. Mengembalikan terkecil (paling dekat dengan infinity negatif) nilai ganda yang lebih besar dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

Kasus khusus : Jika nilai argumen sudah sama dengan bilangan bulat matematika, maka hasilnya adalah sama dengan argumen. Jika argumen adalah NaN atau tak terhingga atau positif nol atau negatif nol, maka hasilnya adalah sama dengan argumen. Jika nilai argumen kurang dari nol tetapi lebih besar dari -1.0, maka hasilnya adalah nol negatif. Perhatikan bahwa nilai Math.ceil (x) adalah persis nilai-Math. Floor(-x).

Parameter: a-nilai.

Pengembalian: yang terkecil (paling dekat dengan infinity negatif) nilai floating-point yang lebih besar dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

✓ **Public Static double floor (double a)**

Menghasilkan bilangan bulat terbesar yang lebih kecil atau sama dengan a. Mengembalikan terbesar (paling dekat dengan infinity positif) nilai ganda yang kurang dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

Kasus khusus : Jika nilai argumen sudah sama dengan bilangan bulat matematika, maka hasilnya adalah sama dengan argumen. Jika argumen adalah NaN atau tak terhingga atau positif nol atau negatif nol, maka hasilnya adalah sama dengan argumen.

Parameter : a- nilai.

Pengembalian: yang terbesar (paling dekat dengan infinity positif) nilai floating-point yang kurang dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

- ✓ **Public Static double exp (double a)**

Menghasilkan angka Euler, e pangkat a. Pengembalian Euler nomor e pangkat dari nilai ganda.

Kasus khusus: Jika argumen adalah NaN, hasilnya adalah NaN.Jika argumen adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argumen adalah infinity negatif, maka hasilnya adalah nol positif. Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat. Hasil harus semi-monoton.

Parameter: a - eksponen untuk meningkatkan e untuk.

Pengembalian : ea nilai, dimana e adalah basis dari logaritma natural.

6) **Public Static double log (double a)**

Menghasilkan logaritma natural dari a. Mengembalikan logaritma natural (base e) dari nilai ganda.

Kasus khusus : Jika argumen adalah NAN atau kurang dari nol, maka hasilnya adalah NAN. Jika argumen adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argumen positif nol atau negatif nol, maka hasilnya adalah tak terhingga negatif. Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat. Hasil harus semi-monoton.

Parameter : a – nilai.

Pengembalian: nilai In a, logaritma natural dari.

- ✓ **Public Static double pow (double a, double b)**

Menghasilkan a pangkat b. Mengembalikan nilai argumen pertama pangkat dari argumen kedua .

Kasus khusus : Jika argumen kedua adalah positif atau negatif nol, maka hasilnya adalah 1,0. Jika argumen kedua adalah 1,0, maka hasilnya adalah sama sebagai argumen pertama. Jika argumen kedua adalah NAN, maka hasilnya adalah NAN.Jika argumen pertama adalah NAN dan argumen kedua adalah nol, maka hasilnya adalah NAN. (Dalam keterangan di atas, nilai floating - point dianggap integer jika dan hanya jika itu adalah terbatas dan titik tetap dari metode ceil atausama titik tetap lantai metode. Nilai A adalah titik tetap dari metode satu - argumen jika dan hanya jika hasil dari penerapan metode untuknilai sama dengan nilai). Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat .Hasil harus semi- monoton.

Parameter: a - dasar . b - eksponen .

Pengembalian :nilai ab .

- ✓ **Public Static long round (double a)**

Menghasilkan pembulatan keatas ke long terdekat. Sebuah method yang di-overload.Dapat juga menggunakan float pada argument dan akan menghasilkan pembulatan ke atas ke int terdekat.

- ✓ **Public Static double sqrt (double a)**

Menghasilkan akar kuadrat a. Mengembalikan bulat benar akar kuadrat positif dari nilai ganda. Kasus khusus:Jika argument adalah NaN atau kurang dari nol, maka hasilnya adalah NaN. Jika argument adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argument positif nol atau negative nol, maka hasilnya adalah sama dengan argumen. Jika tidak, hasilnya adalah nilai ganda yang paling dekat dengan akar kuadrat matematika yang benar dari nilai argumen.

Parameter :a-nilai.

Pengembalian:akar kuadrat positif dari nilai argumen. Jika argument adalah NaN atau kurang dari nol, hasilnya adalah NaN.

- ✓ **Public Static double sin (double a)**

Menghasilkan sinus sudut a dalam radian dan mengembalikan sinus trigonometri dari sudut. Kasus khusus:Jika argumen adalah NaN atau tak terhingga, maka hasilnya adalah NaN. Jika argumen adalah nol, maka hasilnya adalah nol dengan tanda yang sama sebagai argumen. Hasilnya dihitung harus berada dalam 1 penghitungan hasil yang tepat. Hasil harus semi-monoton.

Parameter : a - sudut, dalam radian.

Pengembalian : sinus dari argumen.

- ✓ **Public Static double toDegrees (double angrad)**

Menghasilkan nilai derajat yang kira-kira setara dengan nilai radian yang diberikan. Mengubah sudut diukur dalam radian ke sudut kurang lebih setara diukur dalam derajat. Konversi dari radian ke derajat umumnya eksak, pengguna tidak harus mengharapkan cos (toRadians (90,0)) untuk persis sama 0,0.

Parameter: angrad - sudut, dalam radian.

Pengembalian: pengukuran sudut angrad dalam derajat.

- ✓ **Public Static double toRadians (double angdeg)**

Menghasilkan nilai radian yang kira-kira setara dengan nilai derajat yang diberikan. Dan mengubah sudut diukur dalam derajat ke sudut kurang lebih setara diukur dalam radian. Konversi dari derajat ke radian umumnya eksak.

Parameter: angdeg-sudut, dalam derajat.

Pengembalian: pengukuran angdeg sudut dalam radian.

C. Rangkuman

Ada beberapa class built-in didalam pemrograman java.Yang pertama yaitu class math.Class math dapat berisi method untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma.Selama method-method ini semua static, kita dapat menggunakannya tanpa memerlukan sebuah objek Math.

D. Tugas

Tugas 1

Buatlah program berikut :

Mengambil bilangan yang terbesar dan terkecil dari bilangan berikut :

1 2 3 4 5 6 7 8 9 10

Tugas 2

Buatlah program berikut :

Buat bilangan di bawah ini menjadi acak :

1 2 3 4 5 6 7 8 9 10

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud dengan Class Math?
2. Apa perbedaan dari public static double abs dan public static double random?

G. Lembar Kerja Siswa

5. Kegiatan 5 :Class Built-in (String dan Wrapper)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 5 ini siswa diharapkan dapat :

1) Menganalisis pemanfaatan class String dan Wrapper.
2) Menyajikan beberapa class-class String dan Wrapper dan penerapanya dalam memecahkan masalah.

B. Uraian Materi

1) Class String

Class String disediakan oleh Java SDK dengan menggunakan kombinasi character literals. Tidak seperti bahasa pemrograman lainnya, seperti C atau C++, strings dapat digunakan menggunakan array dari character atau disederhanakan dengan menggunakan class String. Sebagai catatan, bahwa sebuah objek String berbeda dari sebuah array dari character.

✓ **Method-Method String**

- **Public charAt (intidex)**

Mengirim karakter di indeks yang ditentukan oleh parameter index. Mengembalikan nilai arang pada indeks tertentu. Sebuah indeks berkisar dari 0 sampai panjang 1. Nilai Char pertama urutan berada pada indeks 0, berikutnya pada indeks 1, dan seterusnya, seperti untuk pengindeksan array. Jika nilai arang yang ditentukan oleh indeks adalah pengganti, nilai pengganti dikembalikan. Ditentukan oleh: charAt dalam antarmuka CharSequence.

Parameter : Indeks-indeksdari nilai arang.

Pengembalian : nilai arang pada indeks tertentu dari string ini. Nilai Char pertama adalah pada indeks 0. Melempar : Index Out Of Bounds Exception – jika argument indeks negatif atau tidak kurang dari panjang string ini.

- **Public intCompare To (String another String)**

Membandingkan dua String dan mengirim bilangan int yang menentukan apakah objek string pemanggil kurang dari atau sama dengan another String. Bernilai negatif jika objek yang dilewatkan (passed string) lebih besar , 0 jika kedua string sama, dan bernilai positif jika objek string pemanggil (calling string) lebih besar.

Jika tidak ada indeks posisi di mana mereka berbeda, maka string yang lebih pendek leksikografi mendahului lagi tali. Dalam hal ini, compare To mengembalikan perbedaan panjang dari string - yaitu, nilai :

this.length ( ) – another String .length ( )

Ditentukan oleh : compareTo dalam antarmuka Sebanding <String>.

Parameter : anotherString - String yang akan dibandingkan .

Pengembalian : nilai 0 jika argumen string sama dengan string ini, nilai kurang dari 0 jika string ini adalah leksikografis kurang dari argumen string, dan nilai lebih besar dari 0 jika string ini adalah leksikografi lebih besar dari argumen string.

- **Public intCompare To Ignore Case (String str)**

Serupa dengan compare To tetapi case insensitivity. Membandingkan dua string leksikografi, mengabaikan perbedaan kasus. Metode ini mengembalikan sebuah integer yang tanda adalah bahwa memanggil compare To dengan versi normal dari string di mana perbedaan kasus telah dieliminasi dengan memanggil Character.toLowerCase (Character.toUpperCase (karakter)) pada masing-masing karakter. Perhatikan bahwa metode ini tidak mengambil lokal ke account, dan akan menghasilkan pemesanan memuaskan untuk lokal tertentu. Paket java.text menyediakan collators untuk memungkinkan pemesanan lokal-sensitif.

Parameter : str - String yang akan dibandingkan.

Pengembalian : bilangan bulat negatif, nol, atau bilangan bulat positif sebagai String yang ditentukan lebih besar dari, sama dengan, atau kurang dari String ini, mengabaikan pertimbangan kasus.

- **Public Boolean Equals (Object an Object)**

Menghasilkan nilai true jika parameter tunggalnya tersusun dari karakter yang sama dengan objek tempat Anda memanggil equals. Sedangkan jika parameter yang ditentukan bukan sebuah objek String atau jika tidak cocok dengan urutan simbol pada string, method akan dikembalikan dengan nilai false.

- **Public Boolean Equals Ignore Case (String another String)**

Serupa dengan equals tetapi case insensitivity. Membandingkan String ini untuk String lain, mengabaikan pertimbangan kasus. Dua string dianggap sama mengabaikan kasus jika mereka adalah sama panjang dan sesuai karakter

dalam dua string yang sama kasus mengabaikan. Dua karakter c1 dan c2 dianggap mengabaikan kasus yang sama jika setidaknya salah satu dari berikut ini benar : Dua karakter yang sama (dibandingkan dengan operator ==) Menerapkan metode Character.to Upper Case (char) untuk masing-masing karakter menghasilkan hasil yang sama menerapkan metode Character.toLowerCase (char) untuk masing-masing karakter menghasilkan hasil yang sama.

Parameter : anotherString - String untuk membandingkan String ini terhadap Pengembalian : true jika argumen tidak null dan merupakan String mengabaikan kasus setara; false jika tidak.

- **Public Void getChars (int srcBegin, int srcEnd,char[ ] dst, int dst Begin)**

Mendapatkan characters dari string yang dimulai pada index srcBegin hingga index srcEnd dan menyalin karakter-karakter tersebut pada array dst dimulai pada index dstBegin.

- **Public intlength()**

Menghasilkan panjang String. Public String replace (char oldChar, char newChar). Mengganti karakter, semua yang kemunculan oldChar diganti newChar.

- **Public String Substring(int beginIndex, int endIndex)**

Mengirim substring dimulai dari indeks begin Index yang ditentukan dan berakhir dengan indeks endIndex yang ditentukan.

Parameter : beginIndex - indeks awal, inklusif.

Pengembalian : substring yang ditentukan.

Melempar : Index Out Of Bounds Exception - jika beginIndex negatif atau lebih besar dari panjang objek String ini.

- **Public char[] toCharArray()**

Mengembalikan array karakter yang sama dengan string ini.

- **Public StringTrim**

Menghilangkan whitespace di awal dan akhir objek String. Mengembalikan salinan dari string, dengan terkemuka dan trailing spasi dihilangkan. Jika objek String ini merupakan urutan karakter kosong, atau yang pertama dan terakhir karakter dari urutan karakter diwakili oleh objek String ini keduanya memiliki kode lebih besar dari ' \ u0020 ' (karakter spasi), maka referensi ke objek String ini dikembalikan. Jika tidak ada karakter dengan kode yang lebih besar dari ' \ u0020 ' dalam string, maka objek String baru yang mewakili string kosong dibuat dan dikembalikan. Jika tidak, biarkan k menjadi indeks dari karakter pertama dalam string yang lebih besar dari kode ' \ u0020 ', dan biarkan m menjadi indeks dari karakter terakhir dalam string yang lebih besar dari kode ' \ u0020 '. Sebuah objek String baru dibuat, mewakili substring dari string ini yang diawali dengan karakter pada indeks k dan berakhir dengan karakter pada indeks m- yaitu, hasil dari this. Substring (k , m +1). Metode ini dapat digunakan untuk memangkas spasi (seperti dijelaskan di atas) dari awal dan akhir string. Pengembalian : Salinan string ini dengan leading dan trailing spasi dihapus, atau string ini jika tidak memiliki terkemuka atau tertinggal spasi.

- **Public Static StringValueOf(-)**

Dapat menggunakan tipe data sederhana seperti boolean, integer atau character, atau juga menggunakan sebuah objek sebagai parameter. Mengirim objek String yang merepresentasikan tipe tertentu yang dilewatkan sebagai parameter.

2) Class StringBuilder/StringBuffer

Ketika objek String diciptakan, objek String tidak bisa lagi dimodifikasi. Objek StringBuffer serupa dengan objek String, kecuali kenyataan bahwa objek StringBuffer bersifat dapat berubah atau dapat dimodifikasi, sedangkan pada object String bersifat konstan. Dapat dikatakan bahwa class StringBuffer ini lebih fleksibel dibanding class String. Panjang dan isi dapat diubah hingga beberapa pemanggilan method. Class StringBuilder sama dengan class StringBuffer kecuali bahwa metode metode untuk memodifikasi buffer didalam StringBuffer telah disinkronisasi. Class StringBuilder memiliki 3 konstruktor dan

lebih dari 30 metode. StringBuffer dapat digunakan jika diakses oleh beberapa perkerjaan secara bersamaan. Sedangkan StringBuilder digunakan jika diakses oleh satu pekerjaan saja.

3) Class Wrapper

Sesungguhnya, tipe data primitive seperti int, char and long bukanlah sebuah objek. Sehingga, variabel-variabel tipe data ini tidak dapat mengakses method-method dari classObject. Hanya objek-objek nyata, yang dideklarasikan menjadi referensi tipe data, dapat mengakses method-method dari classObject. Ada suatu keadaan, bagaimanapun, ketika Anda membutuhkan sebuah representasi objek untuk variabel-variabel tipe primitive dalam rangka menggunakan method-method Java built-in. Sebagai contoh, Anda boleh menambahkan variable tipe primitif pada objek Collection. Disinilah class wrapper masuk. Class wrapper adalah representasi objek sederhana dari variabel-variable non-objek yang sederhana. Ada 10 tipe data Wrapper, yaitu Boolean, Byte, Character, Double, Float, Integer, Long, Number, Short, dan Void. Perlu diperhatikan bahwa tipe data wrapper dan tipe data dasar (boolean, byte, char, double, float, int, long, short, void) tidak saling menggantikan. Tipe data dasar dilewatkan ke method dengan *pass by value*, jadi jika membutuhkan *pass by reference* harus memanfaatkan kelas tipe data wrapper. Kelas ini menyediakan versi objek dari tipe data dasar, maka dimungkinkan menambah method-method untuk masing-masing tipe.

C. Rangkuman

Class String. Dalam Java, string dapat digunakan menggunakan array dari character atau disederhanakan dengan menggunakan class String. Class StringBuffer, class ini serupa dengan objek string, kecuali kenyataan bahwa objek String Buffer bersifat dapat berubah atau dapat dimodifikasi, sedangkan pada object String bersifat konstan. Class Wrapper, merupakan representasi objek sederhana dari variabel-variabel non-objek yang sederhana.

D. Tugas

Tugas 1

Buatlah program berikut :

Nilai dari variabel angka = 78,6

Buat listing program untuk membuat nilai variabel angka menjadi bilangan bulat. Contoh angka 78,6 dibulatkan ke atas menjadi 79.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari istilah berikut :
   a. Class String
   b. Class StringBuffer
   c. Class Wrapper
2. Sebutkan perbedaan dari StringBuffer dan StringBuilder !
3. Kesalahan apakah yang terdapat pada kode berikut ini ?

Listing Program

```
1 public void Test{
2 String teks;
3 public void Test (String s){
4 this.text = s;
5 }
6 public static void main (String[] args){
7 Test test = new Test (“ABC”);
8 System.out.println(test);
9 }
10 }
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Class String

....

....

....

....

b) Class StringBuffer

....

....

G. Lembar Kerja Siswa

6. Kegiatan 6 : Class Built-in (Class Process dan Class System)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 6 ini siswa diharapkan dapat :

1) Menganalisis pemanfaatan Class Process dan System.
2) Menyajikan beberapa class-class Process dan System dan penerapanya dalam memecahkan masalah.

B. Uraian Materi

1) Class Process

The ProcessBuilder.start ( ) dan metode Runtime.exec membuat proses asli dan mengembalikan sebuah instance dari subclass. Proses yang dapat digunakan untuk mengontrol proses dan mendapatkan informasi tentang hal itu. Class Process menyediakan metode untuk melakukan input dari proses, melakukan output ke proses, menunggu proses untuk menyelesaikan, memeriksa status keluar dari proses, dan menghancurkan (membunuh) proses. Metode yang menciptakan proses mungkin tidak bekerja dengan baik untuk proses khusus pada platform asli tertentu, seperti proses asli windowing, proses daemon, proses Win16/DOS pada Microsoft Windows, atau script shell. Secara default, yang subproses dibuat tidak memiliki terminal atau console sendiri. Yang saya standar / O (yaitu stdin, stdout, stderr) operasi akan diarahkan ke proses induk, di mana mereka dapat diakses melalui sungai diperoleh dengan menggunakan metode getOutputStream ( ), getInputStream ( ), dan getErrorStream ( ). Proses induk menggunakan sungai-sungai ini untuk memberi makan masukan ke dan mendapatkan output dari subproses tersebut. Karena beberapa platform asli hanya menyediakan ukuran buffer terbatas untuk standar input dan output stream, kegagalan untuk segera menulis input stream atau membaca output stream dari subproses dapat menyebabkan subproses untuk memblokir, atau bahkan kebuntuan. Dimana diinginkan, subproses I / O juga dapat diarahkan menggunakan metodedari kelas ProcessBuilder.Subproses tidak tewas ketika tidak ada lagi referensi ke objek Proses, melainkan subproses terus melaksanakan asynchronously. Tidak ada persyaratan bahwa proses diwakili

oleh objek Proses mengeksekusi asynchronous atau bersamaan sehubungan dengan proses Java yang memiliki objek Proses.

2) Method Process

a) Public abstract InputStream getInputStream()

Mengembalikan input stream terhubung ke output normal subproses tersebut. Arus memperoleh data pipa dari output standar dari proses diwakili oleh objek Proses ini. Jika output standar subproses telah diarahkan menggunakan ProcessBuilder.redirectOutput, maka metode ini akan mengembalikan aliran masukan null. Jika standard error sub proses telah diarahkan menggunakan ProcessBuilder.redirectErrorStream, maka input stream dikembalikan oleh metode ini akan menerima output standar digabung dan standard error dari subproses tersebut. Catatan Pelaksanaan : Ini adalah ide yang baik untuk input stream kembali ke buffered. Pengembalian : input stream terhubung ke output normal subproses.

b) Public abstract InputStream getErrorStream()

Mengembalikan input stream terhubung ke output kesalahan subproses tersebut. Arus memperoleh data disalurkan dari kesalahan output dari proses diwakili oleh objek Proses ini. Jika kesalahan standar subproses telah diarahkan menggunakan Process Builder.redirect Error atau Process Builder.redirect Error Stream, maka metode ini akan mengembalikan aliran masukan null. Catatan Pelaksanaan : Ini adalah ide yang baik untuk input stream kembali ke buffered. Pengembalian : Input stream terhubung ke output kesalahan yang subprocess.

3) Class System

Class System menyediakan beberapa field dan method bermanfaat, seperti standard input, standard output dan sebuah method yang berguna untuk mempercepat penyalinan bagian sebuah array.

C. Rangkuman

Class Proccess menyediakan metode untuk melakukan input dari proses, melakukan output ke proses, menunggu proses untuk menyelesaikan, memeriksa status keluar dari proses, dan menghancurkan (membunuh) proses. Method-method yang digunakan pada class Process adalah public abstract InputStream getInputStream() dan public abstract InputStream getErrorStream().

Class System menyediakan beberapa field dan method bermanfaat, seperti standard input, standard output dan sebuah method yang berguna untuk mempercepat penyalinan bagian sebuah array.

D. Tugas

Tugas 1

Buatlah program untuk mengecek inputan suatu bilangan genap,

Contoh : jika inputan itu berupa bilangan ganjil maka akan muncul pesan bahwa inputan salah.

❖ **Mengamati Listing Program dan Output Program**

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari istilah berikut :
   a. Class Process
   b. Class System
2. Pada method process, apa yang dimaksud dengan 'Public Abstract Input Stream getInputStream()' !
3. Apa perbedaan dari getInputStream dan getErrorStream ?

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Class Process

....

....

....

....

b) Class System

....

....

....

....

G. Lembar Kerja Siswa

7. Kegiatan 7 : Exception Handling (Kategori dan Dasar Exception)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 7 ini siswa diharapkan dapat :

1) Mengenal dasar exception handling dan kategori exception.
2) Menyajikan bermacam-macam cara untuk mengimplementasikan kelas RunTimeException.

B. Uraian Materi

1) Dasar Exception

Exception adalah sebuah event yang menjalankan alur proses normal pada program. Event ini biasanya berupa kesalahan(error) dari beberapa bentuk. Ini disebabkan program kita berakhir tidak normal. Dalam bahasa java, ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut exception, yang kemudian dapat diproses lebih lanjut oleh method yang siap menangani kesalahan tersebut. Method tersebut dapat dipilih untuk menangani exception berdasarkan tipe tertentu.Exception dapat muncul tidak beraturan dalam suatu method, atau dapat juga dibuat secara manual dan nantinya melaporkan sejumlah keadaan kesalahan ke method yang memanggil.

2) Tipe-Tipe Exception

Beberapa exception yang telah digunakan dalam bagian-bagian terdahulu adalah ArithmeticException, FileNotFoundException, dan InputMisMatchException. Masih banyak kelas exception lain yang digunakan dalam java, antara lain NullPointerException,ClassNotFoundException,IOException, RunTimeException, IndexOutOBoundsException, IllegalArgumentException, dan masih banyak lagi kelas exception yang digunakan dalam java. Kelas **Throwable** merupakan akar dari semua kelas exception. Semua kelas exception java mewarisi secara langsung atau tidak langsung dari Throwable. Kita bisa menciptakan kelas exception sendiri dengan cara mewarisi exception atau subclass exception. Kelas-kelas exception dapat diklasifikasikan menjadi 3 tipe utama : error system, exception, dan exception runtime.

- Error system dilempar oleh JVM dan direpresentasikan oleh kelas **Error**. Kelas error mendeskripsikan error internal. Error semacam ini jarang terjadi. Jika terjadi, kita dapat memberitahukan kepada user dan menghentikan program secara normal. Contoh sub-kelas Error adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
LinkageError	Kelas punya ketergantungan pada kelas lain, tapi kelas lain telah diubah tanpa menjaga kompatibilitasnya setelah kompilasi dilakukan terhadap kelas pertama.
VirtualMachineError	JVM telah kehabisan sumber daya yang dibutuhkan untuk melanjutkan operasi.

- Exception direpresentasikan dalam kelas **Exception**, yang mendeskripsikan error-error yang diakibatkan oleh program kita dan oleh lingkungan luar. Error ini ditangkap dan ditangani oleh program kita. Beberapa contoh subkelas dari Exception adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
ClassNotFoundException	Percobaan menggunakan suatu kelas yang tidak ada. exception ini terjadi jika kelas yang dijalankan tidak menggunakan perintah java.
IOException	Berkaitan dengan operasi masukan/ keluaran yang tidak valid, membaca melampaui akhir suatu file, dan membuka file yang tidak ada. Subkelas dari IOException antara lain :**InterruptedIOException, EOFException, FileNotFoundException.**

- Exception runtime direpresentasikan oleh kelas **RunTimeException**, yang mendeskripsikan kesalahan pemrograman, seperti casting yang salah, pengaksesan array diluar batas, dan kesalahan numerik. Exception runtime dilempar oleh JVM. Beberapa sub-kelas RunTimeException adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
ArithmeticException	Kesalahan pada operasi aritmatika
IndexOutOfBoundsException	Beberapa jenis indeks diluar batas
NegativeArraySizeException	Array diciptakan dengan ukuran negatif
NullPointerException	Penggunaan acuan null yang tidak valid
ArrayStoreException	Penyimpanan array dengan tipe data yang tidak sesuai
ClassCastException	Cast yang tidak valid
IllegalArrayArgumentException	Argumen yang tidak benar
SecurityException	Aturan sekuriti yang dilanggar
IllegalMonitorStateException	Operasi monitor illegal
IllegalStateException	Lingkungan yang tidak benar
UnsupportedOperationException	Operasi yang tidak didukung

C. Rangkuman

Exception adalah sebuah event yang menjalankan alur proses normal pada program. Event ini biasanya berupa kesalahan error dari beberapa bentuk, disebabkan program kita berakhir tidak normal. Ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut exception. Kelas-kelas exception yang digunakan dalam java adalah ArithmeticException, FileNotFoundException, InputMisMatchException, NullPointerException, ClassNotFoundException, IOException, RunTimeException, dan masih banyak lagi lainnya. Kelas-kelas exception dapat diklarifikasikan menjadi 3 tipe utama yaitu error system, exception, dan exception runtime. Contoh sub kelas error adalah LinkageError, dan VirtualMachineError. Contoh sub kelas exception adalah ClassNotFoundException dan IOException. Dan contoh sub kelas RunTime.

D. Tugas

Tugas 1

Buatlah program sebagai berikut:

Jika script/ listing program dijalankan, akan terdapat pesan kesalahan (kontrol error),

1. NullPointerException
2. ArithmeticException

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari istilah berikut :
   a. Exception
   b. Throwable
   c. IOException

2. Apa kegunaan dari class exception?
3. Sebutkan 4 macam RunTimeException beserta penjelasannya !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a) Exception

....

....

....

....

b) Throwable

…..

....

....

....

c) IOException

....

....

....

....

LJ- 02 : Kegunaan dari class exception :

....

....

....

....

....

....

....

8. Kegiatan 8 : Exception Handling (Exception Handling)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 8 ini siswa diharapkan dapat :

1) Menganalisis mekanisme penanganan kesalahan.
2) Menyajikan bermacam-macam cara untuk mencari tipe kesalahan Uraian Materi.

B. Uraian Materi

1) Exception Handling

Pada dasarnya, *Exception* merupakan subkelas dari kelas *java.lang.Throwable*. "*Bukalah dokumentasi java untuk lebih menyakinkan anda*". Karena *Exception* adalah sebuah kelas maka hakikatnya ketika program berjalan dan muncul sebuah *bug* atau kesalahan, maka *bug* tersebut dapat dianggap sebuah *object*. Sehingga ketika *object* ini di tampilkan di layar maka java akan secara otomatis memanggil method *toString* yang terdapat dalam *object* bertipe *Exception* ini. Java memberikan akses kepada *developer* untuk mengambil *object* bug yang terjadi ini dengan mekanisme yang dikenal *ExceptionHandling*. *Exceptionhandling* merupakan fasilitas di java yang memberikan flexibilitas kepada *developer* untuk menangkap *bug* atau kesalahan yang terjadi ketika program berjalan. Contoh *ExceptionHandling* akan dibahas pada bagian berikutnya.

2) Perbedaan antara ClassError dan ClassException di java

Seperti yang telah dijelaskan di atas bahwa kelas *Exception* merupakan kelas turunan dari kelas *Throwable* di *packageJava.Lang*. Selain *Exception*, java.*lang.Throwable* juga memiliki *subclass* yaitu *classError*. Tentu, kita bertanya-tanya, sebetulnya apa perbedaan antara *classError* dengan *classException*.

3) Penjelasan dari ClassError

"An Error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions." (JDK 5.0 *Documentation*).

4) Penjelasan dari *classException*

"The class *Exception* and its subclasses are a form of *Throwable* that indicates conditions that a reasonable application might want to catch. " (JDK 5.0 *Documentation*) Seperti dari penjelasan yang diberikan oleh *JDKDocumentation*, maka dapat kita lihat bahwa *error* dan *exception* pada dasarnya berbeda. *Error* merupakan masalah yang muncul tapi tidak ada alasan yang kuat untuk menangkapnya. Sedangkan *Exception* merupakan kesalahan kecil yang muncul dan ingin diperlakukan sesuai keinginan developer.

5) Keyword penting pada Exception Handling

Ada 5 *keyword* penting dalam java dalam hal *exception handling* :

a. Try

Keyword try biasanya digunakan dalam suatu *block program*. Keyword ini digunakan untuk mencoba menjalankan *block program*, kemudian mengenai dimana munculnya kesalahan yang ingin diproses. *Keyword* ini juga harus dipasangkan dengan *keyword catch* atau *keyword finally* yang akan dibahas pada point kedua dan ketiga.

Contoh penggunaan :

Listing Program

```
1   public class A
2   {
3   public static void main(String[] args) {
4   try
5   {
6   int a = 1 / 0; // berpotensi untuk menimbulkan
kesalahan yaitu
7   // pembagian dengan bilangan 0
8   System.out.println("perintah selanjutnya");
9   }
10  catch (Exception kesalahan)
11  {
12  System.err.println(kesalahan);
13  }
```

```
14   }
15   }
```

Output :

Output

java.lang.ArithmeticException: / by zero

Perhatikan contoh diatas, ada beberapa hal penting yang perlu dilihat. Pertama, *block program* yang diyakini menimbulkan kesalahan maka ada di dalam *block try and catch*. Kedua, kesalahan yang muncul akan dianggap sebagai *object* dan ditangkap *catch* kemudian di *assign* ke *variable* kesalahan dengan tipe *Exception*. Ketiga, perintah setelah munculnya kesalahan pada *block try* tidak akan dieksekusi.

b. Catch

Jika anda sudah melihat contoh *try* maka secara tidak langsung anda sudah memahami kegunaan dari *keyword* ini. Dalam java, *keyword catch* harus dipasangkan dengan *try*. Kegunaan *keyword* ini adalah menangkap kesalahan atau *bug* yang terjadi dalam *block try.* Setelah menangkap kesalahan yang terjadi maka *developer* dapat melakukan hal apapun pada *block catch* sesuai keinginan *developer*.

Contoh Program :

Listing Program

```
1  catch(Exception kesalahan)
2  {
3  System.out.println("mohon  maaf,  terdapat  kesalahan
pada program");
4  //lakukan hal lainnya disini
5 }
```

Keyword catch juga dapat diletakkan berulang-ulang sesuai dengan kebutuhan.
Contoh :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6 int a = 1/0;  //berpotensi untuk menimbulkan kesalahan
yaitu pembagian dengan bilangan 0
7  System.out.println("perintah selanjutnya");
8  }
9  catch(NullPointerException e)
10 {
11 }
12 catch(ArrayIndexOutOfBoundsException e)
13 {
14 }
15 catch(Exception e)
16 {
17 }
18 }
19 }
```

c. Finally

Keyword finally merupakan *keyword* yang menunjukan bahwa *block program* tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada. Setiap *try* membutuhkan sekurang-kurangnya satu bagian *catch* atau *finally* yang cocok. Jika tidak mendapatkan bagian *catch* yang cocok, maka bagian *finally* akan dieksekusi sebelum akhir program, atau setiap kali suatu *method* akan kembali ke pemanggilnya, melalui *exception* yang tidak dapat ditangkap, atau melalu pernyataan *return*, bagian *finally* akan dieksekusi sebelum kembali ke *method* lagi.

Contoh implementasinya pada program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  int a = 1/0; }
7  finally
8  {
9  System.out.println("terima kasih telah menjalankan
program");
10  }
11  }
12  }
```

Output Program diatas:

Output

terima kasih telah menjalankan program

Jika saya lakukan modifikasi program diatas menjadi :

Listing Program

```
1   public class A
2   {
3   public static void main(String[] args) {
4   try
5   {
6   int a = 1/0;
7   }
8   catch (Exception e)
9   {
10  System.out.println(“ada kesalahan yang muncul”);
11  }
```

```
12  finally
13  {
14   System.out.println(“terima  kasih  telah  menjalankan
program”);
15  }
16  }
17  }
```

Output Program :

Output

ada kesalahan yang munculterima kasih telah menjalankan program

Perhatikan kedua contoh diatas, *block finally* akan selalu dieksekusi meskipun adanya kesalahan atau tidak pada *block try*. Berbeda dengan *keyword catch*, *keyword finally* hanya dapat diletakkan 1 kali setelah *keyword try*.

d. Throw

Keyword Throw digunakan untuk melemparkan suatu *bug* yang dibuat secara manual.

Contoh program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  throw new Exception("kesalahan terjadi");
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
```

```
13 }
```

Output Program :

Output

java.lang.Exception: kesalahan terjadi

Seperti yang anda lihat pada program diatas, pada *keyword* **throw new Exception ("kesalahan terjadi");** akan melempar *object* bertipe *exception* yang merupakan *subclass* dari *class Exception* sehingga akan dianggap sebagai suatu kesalahan yang harus ditangkap oleh *keyword catch*.

Perhatikan contoh berikut ini :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6   throw new B(); //cobalah  ganti baris ini dengan à
new B();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 }
14 class B extends Exception
15 {
16 B()
17 {
18 }
```

```
19 public String toString()
20 {
21 return “object dengan tipe kelas B”;
22 }
23 }
```

Output Program :

Output

object dengan tipe kelas B

Program diatas telah mendefinisikan suatu kelas B *mengextends* dari kelas *Exception*. Ketika kita melakukan *thrownewB();* maka object dari kelas bertipe B ini akan dianggap kesalahan dan ditangkap oleh *blockcatch*. Sekarang jika anda menghilangkan keywordthrow apa yang terjadi?

.

e. Throws

Keywordthrows digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya.Cara mendefinisikannya dalam method adalah sebagai berikut : *<method modifier> type method-name throws* **exception-list1, exceptio-list2, …** {}.

Contoh Program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  f();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
```

```
12 }
13 public static void f() throws NullPointerException,
ArrayIndexOutOfBoundsException
14 {
15 //implementasi method
16 throw new NullPointerException();
17 //throw new ArrayIndexOutOfBoundsException();
18 }
19 }
```

Output Program :

Output

java.lang.NullPointerException

Contoh program lainnya :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  f();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 public static void f() throws NullPointerException,
ArrayIndexOutOfBoundsException
14 {
15 //implementasi method
16 //throw new NullPointerException();
```

```
17 throw new ArrayIndexOutOfBoundsException();
18 }
19 }
```

Output Program :

Output
java.lang.ArrayIndexOutOfBoundsException

Perhatikan kedua contoh pengguaan keyword*throws* pada *method*. Ketika method tersebut dipanggil dalam *blocktry*. Maka method tersebut akan membuat *object* yang merupakan *subclass* dari *classThrowable* dan methodtersebut akan melemparkan kesalahan yang ada dalam *blockmethod* kedalam *blocktry*. Di dalam *blocktry*, kesalahan tersebut kemudian ditangkap kedalam *blockcatch*.

C. Rangkuman

Java memberikan akses kepada developer untuk mengambil object bug yang terjadi dengan mekanisme yang dikenal dengan ExceptionHandling. ExceptionHandling merupakan fasilitas di java yang memberikan flexibiitas kepada developer untuk menangkap bug atau kesalahan yang terjadi ketika program berjalan. Dalam exception Handling ada 5 keyword penting yang digunakan, yaitu Try yang digunakan untuk menjalankan block program kemudian mengenai dimana kesalahan yang akan diproses. Catch digunakan berpasangan dengan keyword try, keyword ini digunakan untuk menangkap kesalahan yang terjadi dalam block try. Finally digunakan untuk menunjukkan bahwa block program akan selalu dieksekusi meskipun adanya kesalahan yang muncul ataupun tidak ada. Keyword throw digunakan untuk melemparkan bug yang dibuat secara manual, keyword ini digunakan dalam satu method atau kelas yang menghasilkan kesalahan sehingga perlu ditangkap errornya.

D. Tugas

Tugas 1

Tulislah suatu program yang meminta user untuk memasukkan 2 integer dan menampilkan penjumlahan atas keduanya. Program anda harus meminta pengguna memasukkan 2 integer kembali bila inputan sebelumnya tidak tepat. Tipe exception yang digunakan adalah **NumberFormatException**.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   a. Try
   b. Catch
   c. Finally
2. Apa kegunaan dari keyword Throw dan Throws ?

3. Bagaimana cara melemparkan suatu exception? Apakah boleh melempar beberapa exception sekaligus menggunakan satu statement **throw**?

4. Apa keluaran (output) dari kode berikut ini ?

Listing Program

```
1 public class Test{
2public static void main (String [] args){
3  try{
4  int nilai = 30;
5  if (nilai < 40)
6  throw new Exception ("nilai terlalu kecil");
7  }
8  catch (Exception ex){
9  System.out.println (ex.getMessage());
10 }
11  System.out.println  ("Lanjut  setelah  blok
catch");
12  }
13  }
```

Apakah keluaran kode bila baris

`Int nilai = 30 ;` Diganti dengan `Int nilai = 50 ;`

5. Jelaskan perbedaan antara catch dan throw !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a)Try

....

....

....

....

9. Kegiatan 9 : Exception Handling (Penanganan Exception)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 9 ini siswa diharapkan dapat :

1) Menganalisis aturan penanganan Exception.
2) Menyajikan bermacam-macam cara untuk menangani masalah exception handling.

B. Uraian Materi

1) Penanganan Exceptions

Untuk menangani exception dalam java, kita gunakan blok try-catch-throw-throws-finally. Apa yang kita lakukan dalam program kita adalah kita menempatkan pernyataan yang mungkin menghasilkan exception dalam blok ini. Bentuk umum dari blok try-catch-finally adalah :

Listing Program

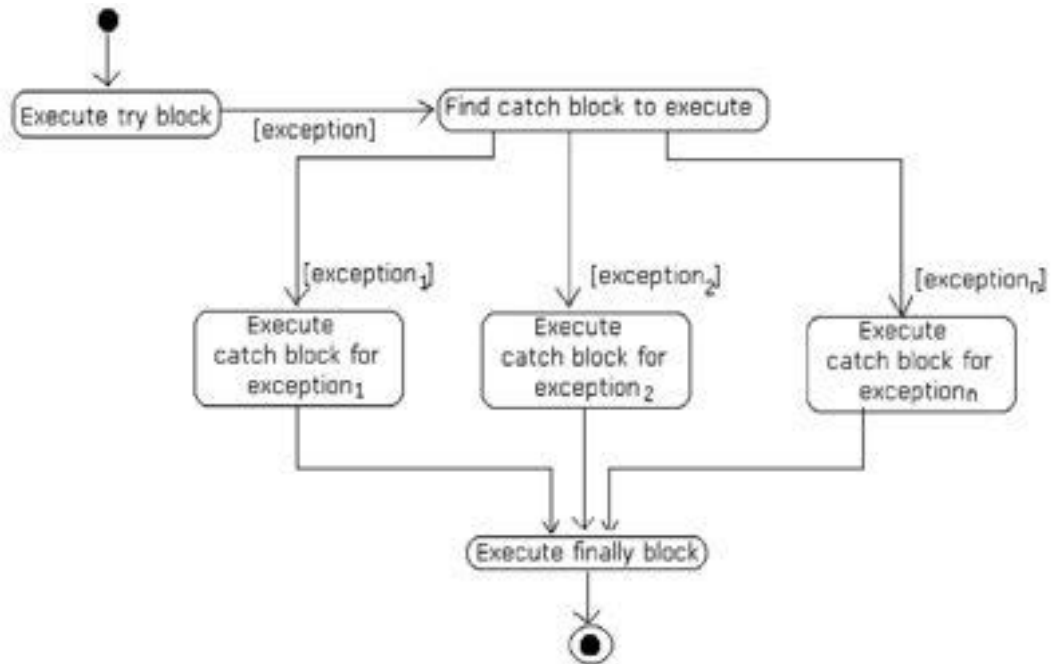
```
1   try{
2   //tulis pernyataan yang dapat mengakibatkan exception
3   //dalam blok ini
4   }
5   catch( <exceptionType1><varName1> ){
6    //tulis aksi apa dari program Anda yang dijalankan
jika ada
7   //exception tipe tertentu terjadi
8   }
```

Exception dilemparkan selama eksekusi dari blok *try* dapat ditangkap dan ditangani dalam blok *catch*. Kode dalam blok *finally* selalu di-eksekusi.

Berikut ini adalah aspek kunci tentang sintak dari konstruksi try-catch-finally:

1. Notasi blok bersifat perintah.
2. Setiap blok *try*, terdapat satu atau lebih blok *catch*, tetapi hanya satu blok *finally*.
3. Blok *catch* dan blok *finally* harus selalu muncul dalam konjungsi dengan blok *try*, dan di atas urutan.

4. Blok *try* harus diikuti oleh **paling sedikit** satu blok *catch* ATAU satu blok *finally,* atau keduanya.
5. Setiap blok *catch* mendefinisikan sebuah penanganan exception. Header dari blok *catch* harus membawa satu argumen, dimana exception pada blok tersebut akan ditangani.

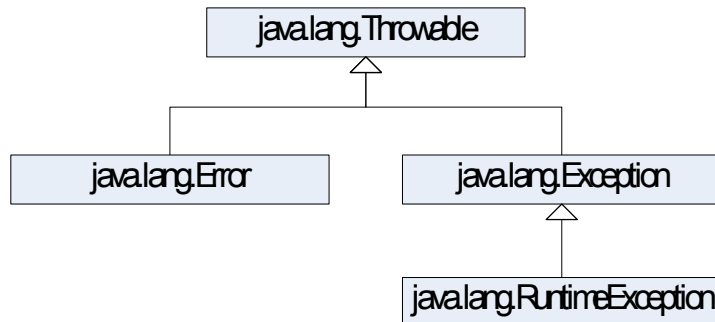
Exception harus menjadi class pelempar atau satu dari subclassesnya.

Ada lima keywords yang digunakan oleh Java untuk menangani exception ini, yaitu :***Try*, *catch*, *finally*, *throw*, *throws*.**

Semua class exception terdapat dalam package **java.lang**.Superclass tertinggi adalah class **Throwable**, tetapi kita hampir tidak pernah menggunakan class ini secara langsung.

Class **Error→** tipe exception yang seharusnya tidak ditangani dengan menggunakan blok try catch karena berhubungan dengan Java run-time system/evironment. Jadi, exception yang terjadi kemungkinannya sangat kritis yang sebaiknya tidak ditangani oleh program kita sendiri.

Class **Exception→** tipe exception yang sebaiknya ditangani oleh program kita secara langsung.

Dalam penggunaannya, kita akan banyak menangani exception yang merupakan turunan dari class **Exception** ini. Salah satu turunannya yang perlu diperhatikan

adalah class **RuntimeException**, karena Java memperlakukan class ini & turunannya secara berbeda.

2) Menampilkan Pesan Exception

Beberapa method standard yang dapat digunakan untuk menampilkan pesan exception merupakan anggota dari kelas java.lang.Throwable.

No	Method Pesan Exception	Deskripsi
1	getMessage()	Mengembalikan nilai string yang berisi pesan rinci tentang objek Throwable yang mengalami exception
2	toString()	Mengembalikan nilai string yang berisi pesan singkat tentang objek yang mengalami exception
3	getLocalizedMessage()	Menampilkan pesan exception lokal (yang terjadi pada subkelas saja)
4	printStackTrace()	Method ini bersifat void, dan hanya mencetak informasi tentang objek Throwable

C. Rangkuman

Untuk menangani exception, dapat menggunakan blok try-catch-throw-finally. Exception dilemparkan selama eksekusi dari blok try dapat ditangkap dan ditangani dalam blok catch. Kode dalam blok finally selalu dieksekusi. Semua class exception terdapat dalam package java.lang. Superclass tertinggi adalah class Throwable, tetapi kelas ini jarang digunakan.

D. Tugas

Tugas 1

Buatlah program dengan blok try-catch-thow-finally dimana exception memuat kesalahan ArithmeticException, dimana program akan menampilkan kesalahan lokal yang termuat!

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   a. Try
   b. Catch
   c. Finally
2. Apa kegunaan dari keyword Throw dan Throws ?
3. Bagaimana cara melemparkan suatu exception? Apakah boleh melempar beberapa exception sekaligus menggunakan satu statement **throw**?

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a)Try

....

....

....

....

b) Catch

…...

....

....

....

c) Finally

....

....

....

....

G. Lembar Kerja Siswa

10. Kegiatan 10 : String (Pengantar dan Penyimpanan String)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 10 ini siswa diharapkan dapat :

1) Memahami tentang pengertian String.
2) Menyajikan hasil dari pengolahan String.

B. Uraian Materi

1) Pengolahan String

Dalam pemrograman Java string merupakan aspek penting, karena dapatmempelajari mengenai class dan objek melalui penggunaan string. String sebenarnya merupakan class yang terdapat dalam library Java. Java String merupakan salah satukelas dasar yang disediakan oleh Java untuk memanipulasi karakter.

2) Membuat Objek String

Java mendefinisikan class String dalam package java.lang.String, sehingga tidak perlu melakukan impor secara eksplisit. Java String digunakan untuk mendefinisikan string yang konstant (tidak bisa berubah).

Contoh dasar penggunaan String pada Java :

Listing Program

```
1  public class ST
2  {
3  public static void main (String args []) {
4  String str1 = "contoh string";
5  System.out.println(str1);
6  }
7  }
```

3) String Immutability

String merupakan class yang immutable. Ini berarti isi dari String ini tidak dapat diubah lagi ketika ia sudah terbentuk. Jika kita mendefinsikan isi string seperti di bawah ini.

String string = new String(“Rhamdani”);

Kita tidak akan dapat mengubah isi string seperti di bawah ini:

String substring (3,string.length());

Isi string tetap sama, yaitu Rhamdani. Untuk dapat mengakalinya, kita harus menangkap operasi substring tersebut ke dalam sebuah string lainnya atau bisa saja ke dalam string itu sendiri. Di bawah ini contohya:

Listing Program

```
1 //menangkap hasil operasi substring ke dalam string lain
2    String    stringBAru    =    string.    Substring
(3,string.length());
3  //menangkap hasil operasi substring ke dalam string itu
   sendiri
4  String = string.substring(3,string.length());
```

Untuk catatan, operasi di atas tidak hanya berlaku pada substring() saja. Tapi pada semua operasi pada class string.

4) Menggabungkan String

Seringkali dalam pemrograman kita perlu menggabungkan String untuk mendapatkan String baru. Kita dapat menggunakan operator (+) untuk menggabungkan beberapa String.

Contoh penggunaan :

Listing Program

```
1 public class ST{
2 public static void main (String args[]){
3 String kata1 = "ini contoh string";
4 String kata2 = "pada java";
5 System.out.println(kata1 + kata2);
6 }
7 }
```

C. Rangkuman

Dalam pemrograman java, String merupakan aspek penting,karena mempelajari class dan objek melalui penggunaan String. String adalah class yang terdapat dalam library Java. Java String merupakan salah satu kelas dasar yang disediakan untuk memanipulasi karakter. Definisi String pada Java yaitu java.lang.String, sehingga tidak perlu import secara eksplisit. String merupakan class yang immutable, yang berarti isi dari String tidak dapat diubah ketika sudah terbentuk.

D. Tugas

Tugas 1

Buatlah sebuah program dimana program berisi penggabungan 2 string, misal:

String 1 : “Bermain”

String 2 : “Bola”

Tampilan pada layar : “Bermain Bola”

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa fungsi dari String?
2. Apa yang dimaksud dengan String Immutability?
3. Apakah output dari kode berikut ini !

Listing Program

```
1 String str1 = “latihan belajar java”;
2 String str2 = “di dalam kelas”;
3 System.out.println (str1 + ”bersama teman sekelas” + str2);
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Fungsi dari String :

....

....

....

....

....

G. Lembar Kerja Siswa

11. Kegiatan 11 : String (String Immutability dan StringBuffer Class)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 11 ini siswa diharapkan dapat :

1) Memahami tentang pengertian StringBuffer.
2) Menyajikan hasil dari pengolahan String.

B. Uraian Materi

1) StringBuffer

StringBuffer adalah pasangan class String yang menyediakan banyak fungsi string yang umum. StringBuffer merepresentasikan urutan karakter yang dapat dikembangkan dan ditulis ulang. StringBuffer dapat disisipi karakter dan subString di tengahnya, atau ditambah di belakangnya. StringBuffer memiliki default kapasitas 16 karakter, tapi biasanya ukuran diatur sendiri dengan mendefinisikan kapasitas pada saat pembuatan.

Misalnya adalah sebagai berikut:StringBuffer baru = new StringBuffer(50); Contoh diatas merupakan StringBuffer kosong dengan kapasitas 50 karakter. Ada 3 cara untuk mengefinisikan StringBuffer :

- **StringBuffer baru = new StringBuffer()**

Secara tidak langsung variabel baru akan menjadi objek StringBuffer dengan ukuran 16 karakter karena defaultnya adalah 16 karakter

- **StringBuffer baru1 = new StringBuffer(50)**

Objek baru1 merupakan StringBuffer dengan panjang karakter 50

- **StringBuffer baru2 = new StringBuffer(*String*)**

Objek baru2 merupakan objek StringBuffer dengan panjang karakter ***String* + 16 karakter.**

Berikut merupakan contoh deklarasi StringBuffer :

Listing Program

```
1  public class SB{
2  public static void main (String args[]){
3  String kata = "Java";
4  StringBuffer baru = new StringBuffer();
5  StringBuffer baru1 = new StringBuffer(50);
6  StringBuffer baru2 = new StringBuffer(kata);
7  System.out.println("baru     : "+baru.capacity());
8  System.out.println("baru1    : "+baru1.capacity());
9  System.out.println("baru1    : "+baru2.capacity());
10 }
11 }
```

Berbeda denga type data primitif yang lain, type data String dalam Java diperlakukan sebagai object. Object String dalam Java dapat dibuat dengan dua cara, yaitu:

Penulisan sesuatu di dalam tanda antara petik ganda (literal String). Cara ini digunakan untuk mengakomodasi kebiasaan dari bahasa C/C++

- String s = ”Hello World”;
- System.out.println(“Hello World”);

Pembuatan object String dengan keyword new.

- String s = new String(“Halo”);

Class String mempunyai atribut final, sehingga Class String tidak dapat di-extends / diturunkan.

Dalam Java, terdapat dua jenis memory yaitu:

- Stack (tempat local variable dan tumpukan method)
- Heap (tempat instance variable dan object), Di dalam heap terdapat bagian memory yang disebut dengan **String constant pool.**

Bila kita membuat object String dengan penulisan sesuatu di antara tanda petik ganda (literal String), maka object String tersebut akan berada di dalam String constant pool. Sedangkan bila kita membuat String dengan keyword new, maka object String tersebut akan berada di dalam heap (tetapi diluar String constant pool). Khusus untuk pembuatan object String dengan keyword **new**

(ex: String sample = new String("Hello World")), sebenarnya terdiri dari 3 buah proses (yang melibatkan 2 buah object String), yaitu :

- Pembuatan object String "Hello World" di dalam String constant pool. Hal ini karena "Hello World" adalah literal String yang otomatis membuat object di String constant pool.
- Pembuatan object String "Hello World" di dalam heap (non constant pool).
- Penghapusan object String "Hello World" di dalam String constant pool (bila tidak ada yang mereferensi String) ini.

Pada saat java menjumpai literal String (dalam kode program), maka java akan mencari String yang sama dengan literal String tersebut di dalam String constant pool. Bila ternyata di dalam pool ditemukan object String yang sama, maka reference akan menunjuk pada object String di dalam pool tersebut. Bila ternyata java tidak menemukan di dalam String constant pool, maka java akan membuat object String di dalam String constant pool terlebih dahulu.
Object String adalah immutable (tidak dapat diubah) . Pengertian tidak dapat diubah adalah, sekali sebuah object String berisi suatu nilai, maka nilai tersebut tidak dapat diubah (tidak peduli apakah object String tersebut berada pada heap ataupun String constant pool). Dalam praktek pemrograman, kita merasa bahwa object String dapat berubah, hal ini karena yang berubah adalah nilai reference penunjuk object String bukan object String tersebut.

String Class sangat tidak efektif bila kita ingin melakukan banyak modifikasi terhadap suatu String object, hal ini karena sifat dari String Class yang immutable (banyak modfikasi pada suatu kelas String akan dapat menyebabkan banyaknya object String yang terlibat). StringBuffer dan StringBuilder Class mengatasi permasalahan ini (mutable). StringBuffer thread safe sehingga dapat menjamin konsistensi operasi pada String object.

Pada String Class, method-method akan mengembalikan object String baru (hasil modifikasi) tanpa mengubah object String tempat method dipanggil (karena immutable)Pada kelas StringBuffer Class method-method akan memodifikasi object tempat method dipanggil, dan kemudian mengembalikan object tersebut sebagai return value dari method.Method utama pada StringBuffer adalah

append dan insert method.Dan dengan Class StringBuffer kita dapat melakukan chaining method.

2) Konstruktor

Konstruktor pada Java merupakan method khusus yang dipakai oleh Java untuk membuat sebuah object didalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor.

Karakteristik konstruktor :

a) Nama Konstruktor = Nama Kelas.
b) Tidak mengembalikan nilai atau return value termasuk void.
c) Cara menggunakan konstruktor adalah dengan menggunakan kata kunci **new**. Jika didalam kelas tidak dituliskan konstruktor, Java akan secara default menambahkan konstruktor kosong kedalam kelas tersebut.

3) Pemanggilan Konstruktor

Membuat konstruktor

Listing Program

```
1  public class Constr{
2  Constr ()
3  {
4  }
5  }
```

Konstruktor dipanggil saat membuat sebuah object. Sama seperti membuat object pada class.

Contoh :

Listing Program

```
1  public class Constructor_1{
2  float nilaiAkhir;
3  public Constructor_1(int nilai_akhir){
4  nilaiAkhir=nilai_akhir;
5  }
6  public String grade(){
```

```
7  String nilaigrade;
8  if(nilaiAkhir>=50)
9  nilaigrade="Bagus";
10 else
11 nilaigrade="Jelek";
12 return nilaigrade;
13 }
14 public void cetak(){
15 System.out.println("Grade nilainya = "+grade());
16 }
17 public static void main (String [] args){
18 Constructor_1 hasil = new Constructor_1(67);
19 hasil.cetak();
20 }
21 }
```

4) Overload Konstruktor

Sebuah class mungkin memiliki lebih dari satu konstruktor dengan parameter yang berbeda satu sama lainnya.

Contoh :

Listing Program

```
1  public class Constructor_2{
2  float nilai1, nilai2;
3  public Constructor_2(int bil1){
4  nilai1 = bil1;
5  }
6  public Constructor_2(int bil1, bil2){
7  nilai = bil1+bil2;
8  }
9  public String grade()
10 {
11 String nilaigrade;
12 if(nilai1>=50)
13 nilaigrade = "Bagus";
```

```
14 else
15 nilai grade = "Jelek";
16 return nilaigrade;
17 }
18 public void cetak(){
19 System.out.println("Grade nilainya = "+grade());
20 }
21 public static void main (String [] args){
22 Constructor_2 hasil = new Constructor_2 (45);
23 hasil.cetak();
24 Constructor_2 hasilnya = new Constructor_2 (45,35);
25 hasilnya.cetak();
26 }
27 }
```

3) Modifier

Modifier digunakan untuk menentukan sifat dari suatu kelas dan menentukan preveledge (hak akses) dari kelas lain. Modifier juga digunakan untuk menentukan relasi (extend atau implements) dengan kelas lainnya. Berikut ini adalah wilayah modifier akses :

Wilayah Akses	Public	Protected	Default	Private
Dikelas yang sama	✓	✓	✓	✓
Beda kelas, di package yang sama	✓	✓	✓	✘
Beda kelas, beda package, dikelas turunan	✓	✓	✘	✘
Beda kelas, beda package, tidak di kelas turunan	✓	✘	✘	✘

Keterangan :

1. Public : menyatakan bahwa kelas/ method/ attribute dapat diakses oleh kelas lain dimanapun letaknya.
2. Protected : menyatakan bahwa kelas/ method/ attribute tersebut dapat diakses oleh kelas lain yang berada dalam satu package atau kelas lain tersebut merupakan turunannya.
3. Private : menyatakan bahwa kelas tersebut tidak dapat diakses sama sekali oleh kelas lain bahkan tidak dapat diakses oleh kelas turunannya. Attribute yang bersifat private hanya dapat diakses oleh method dalam kelas yang sama, kelas lain masih dapat mengakses melalui method tersebut asal method tersebut modifiernya public.

4) Class String

String pada java adalah object dan sifatnya read-only (immutable).Karena sifat immutable ini setiap perubahan terhadap isi string akan dibuat string baru untuk menampung perubahan tersebut.

Contoh :

String s1 = “hello world”;

S1 = “hello java”;

Akan menyebabkan dua kali pengalokasian memory untuk dua objek string tersebut diatas, dengan referensi terakhir s1 ke string “hello java”.

5) Method pada class String

Method adalah bagian-bagian kode yang dapat dipanggil ole program utama atau dari method lainnya untuk menjalankan fungsi yang spesifik.

karakteristik dari method diantaranya sebagai berikut :

a) Dapat mengembalikan satu nilai atau tidak sama sekali
b) Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada paramater sama sekali. Parameter bisa juga disebut sebagai argumen fungsi.
c) Setelah method telah selasai dieksekusi, method akan kembali pada method yang memanggilnya.

- **Menentukan Awal Dan Akhir String**

Untuk menentukan awal dan akhir String, kita dapat menggunakan dua fungsi utama, yaitu : startsWith(String s)
Dengan fungsi ini, maka objek String yang bersangkutan akan diperiksa,
apakah diawali oleh objek String s, pada parameter fungsi ini endsWith(String s)
Dengan fungsi ini, maka objek string yang bersangkutan akan diperiksa,
apakah diakhiri oleh objek string s, pada parameter fungsi ini.
Fungsi diatas akan menghasilkan nilai boolean ***true*** bila benar dan ***false*** bila salah.

- **Mengurutkan String**

Dapat juga melakukan pengurutan string dengan method compareTo().Method ini membandingkan karakter-karakter pada String secara berurutan dari awal String. Misalnya string pertama bernilai “a” dan string kedua bernilai “b”, maka apabila Stringpertama.compareTo(String kedua) akan menghasilkan nilai negatif (<0)dan apabila dilakukan sebaliknya akan menghasilkan nilai positif (>0). Nilai 0 akan dihasilkan apabila string pertama dan kedua sama.
Contoh penggunaannya:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String kata = “imam”;
4  String kata2 = “agung”;
5  String kata3 = “uding”;
6  System.out.println(kata2.compareTo(kata3));
7  System.out.println(kata.compareTo(kata2));
8  }
9  }
```

- **Mendapatkan Panjang String**

Kita dapat memperoleh panjang string dengan menggunakan method length(); seperti contoh berikut ini :

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String panjang = "panjang string";
4  System out.println(panjang.length());
5  }
6  }
```

- **Mencari Karakter Pada Index Tertentu**

Kita bisa mencari tahu karakter apa yang ada pada index tertentu yang terdapat pada String.

Contoh:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String str01 = "contoh string";
4  System.out.println(str01.charAt(0));
5  }
6  }
```

- **Mencari Posisi Karater Atau SubString Dari String**

Ada dua method yang dapat digunakan untuk mencari posisi karakter dari string dan dua method untuk mendapatkan posisi subString dari string.

- **Method untuk mencari posisi karakter pada String :**

IndexOfchar (karakter) Memerlukan argumen berupa karakter dan akan mengembalikan nilai posisi indeks dari karakter yang dicari. Posisi yang dikembalikan adalah posisi pertama dari karakter yang ditemukan. Bila karakter tidak ditemukan, maka akan mengembalikan nilai -1.IndexOf(char karakter, int indeks) *s*ama dengan sebelumnya, tetapi memerlukan argumen tambahan, yaitu indeks posisi awal pencarian dalam integer.

Contoh:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String str01 = "contoh string";
4  System.out.println(str01.indexOf("h"));
5  System.out.println(str01.indexOf("h", 7));
6  }
7  }
```

- **Method untuk mencari posisi subString pada String :**

indexOf(String Str)Penggunaan dan fungsi sama dengan method untuk char.

indexOf(String str, int indeks) Penggunaan dan fungsi sama dengan method untuk char.

- **Method untuk mengubah ke huruf besar semua**

Penggunaan method ini berfungsi untuh mengubah huruf pada suatu string menjadi huruf besar semua.

Contoh:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String kalimat = new String("saya pasti bisa");
4  System.out.println(kalimat.toUpperCase());
5  }
6  }
```

- **Method untuk mengubah ke huruf kecil semua**

Penggunaan method ini berfungsi untuh mengubah huruf pada suatu string menjadi huruf kecil semua.

Contoh:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String kalimat = new String("SAYA PASTI BISA");
4  System.out.println(kalimat.toLowerCase());
5  }
6  }
```

- **Method untuk mengganti huruf tertentu**

Penggunaan method ini berfungsi untuh menganti char tertentu dengan char yang baru, jadi nantinya akan muncul String yang baru.

Contoh:

Listing Program

```
1  public class ST{
2  public static void main (String args[]){
3  String kata= "malam";
4  System.out.println(kata.replace("l", "k"));
5  }
6  }
```

C. Rangkuman

StringBuffer adalah pasangan class String yang menyediakan banyak fungsi string yang umum. StringBuffer merepresentasikan urutan karakter yang dapat dikembangkan dan ditulis ulang. StringBuffer dapat disisipi karakter dan substring ditengah atau dibelakangnya. Panjang karakter String yaitu 16 karakter.

Pada String Class, method-method akan mengembalikan object String baru tanpa mengubah object String tempat method dipanggil (karena immutable) pada

kelass StringBuffer method-method akan dimodifikasi object tempat method dipanggil, kemudian mengembalikan object sebagai return value dari method.
Konstruktor merupakan method khusus yang dipakai oleh java untuk membuat sebuah object didalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor.
Modifier digunakan untuk menentukan sifat dari suatu kelas dan menentukan preveledge dari kelas lain. Modifier juga digunakan untuk menentukan relasi dengan kelas lainnya.

C. Tugas

Tugas 1

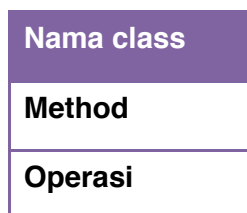
Buatlah program untuk mengurutkan kata :

```
Pemrograman belajar itu java menyenangkan
```

Menjadi

```
belajar pemrograman java itu menyenangkan
```

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

D. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud StringBuffer?
2. Sebutkan minimal 5 method pada String dan jelaskan fungsinya !

F. Lembar Kerja Siswa

12. Kegiatan 12 : String (Constructor, Method dan Class StringBuffer)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 12 ini siswa diharapkan dapat :

1. Memahami constructor , method dan class stringBuffer
2. Menerapkannya pada suatu program

B. Uraian Materi

1. Constructor StringBuffer

StringBufer mempunyai tiga buah constructor. Kita dapat mengkonstruksi obyek kelas StringBuffer menggunakan salah satu di antara tiga konstruktor. Berikut ini adalah tiga constructor StringBuffer:

- **StringBuffer()**

Untuk mengkonstruksi buffer string kosong (tanpa karakter).

- **StringBuffer(int length)**

Untuk mengkonstruksi buffer string kosong (tanpa karakter) dengan kapasitas ditentukan oleh parameter lengh.

- **StringBuffer(String str)**

Untuk mengkonstruksi buffer string dengan parameter String.

2. Beberapa Method pada class StringBuffer

- **Capacity()**

Method ini digunakan untuk mengetahui kapasitas dari suatu variable StringBuffer. Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "Java";
4  StringBuffer baru = new StringBuffer();
5  StringBuffer baru1 = new StringBuffer(50);
6  StringBuffer baru2 = new StringBuffer(kata);
```

```
7  System.out.println("baru    : "+baru.capacity());
8  System.out.println("baru1   : "+baru1.capacity());
9  System.out.println("baru1   : "+baru2.capacity());
10 }
11 }
```

- **Reverse()**

  Method ini digunakan untuk membalik isi dari variable StringBuffer.

  Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println("baru awal       : "+baru2);
6  System.out.println  ("baru  reverse()  :  "+baru2.reverse
   ());
7  }
8  }
```

- **setCharAt(,)**

  Method ini digunakan untuk mengubah karaker pada indeks tertentu. Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println("baru awal       : "+baru2);
6  baru2.setCharAt(0,'s'); //mengubah S menjadi s
7  baru2.setCharAt(6,'b'); //mengubah B menjadi b
8  System.out.println("baru charAt1       : "+baru2);
9  }
```

```
10 }
```

- **append()**

  Method ini digunakan untuk menambahkan string pada akhir StringBuffer. Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println ("\nbaru (append): "+baru2.append ("
   method append"));
6  }
7  }
```

- **insert(,)**

  Method ini digunakan untuk menyisipkan string pada posisi tertentu. Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println  ("\nbaru  (insert):  "+baru2.insert
   (6, " dan "));
6  }
7  }
```

- **delete(,)**

  Method ini digunakan untuk menghapus string pada indeks tertentu. Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println("\nbaru (awal)  : "+baru2);
6  System.out.println  ("baru  (delete):  "+baru2.delete
   (4,8));
7  }
8  }
```

- **Length()**

  Method ini digunakan untuk mengetahui panjang objek.

  Contoh:

Listing Program

```
1  public class modul{
2  public static void main (String args[]){
3  String kata = "StringBuffer";
4  StringBuffer baru2 = new StringBuffer(kata);
5  System.out.println        ("\nbaru        (capacity):
   "+baru2.capacity ());
6  System.out.println  ("baru  (length)   :  "+baru2.length
   ());
7  }
8  }
```

C. Rangkuman

StringBuffer mempunyai tiga buah konstruktor yaitu StringBuffer () yang digunakan untuk mengkonstruksi buffer string kosong, StringBuffer (int leght) digunakan untuk mengkonstruksi buffer string kosong (tanpa karakter) dan kapasitas ditentukan oleh parameter length, dan yang terakhir StringBuffer (String str) digunakan untuk mengkonstruksi buffer string dengan parameter String.

Beberapa method pada class StringBuffer yaitu Capasity(), Reverse(), setCharAt(,), append(), insert(), delete(), dan length().

D. Tugas

Tugas 1

Buatlah suatu program untuk menyisipkan kata dalam sebuah kalimat, misalnya:

```
Saya sedang belajar Java
```

Sisipkan kata pemrograman diantara kata belajar dan java

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa fungsi dari konstruktor dan method?
2. Sebutkan minimal 5 method pada StringBuffer dan jelaskan fungsinya !
3. Sebutkan konstruktor yang ada pada StringBuffer dan jelaskan fungsinya !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Fungsi dari konstruktor dan method :

....
....
....
....
....

LJ- 02 : 5 Method pada StringBuffer dan fungsinnya :

....
....
....
....
....
....
....
....
....

LJ- 03 : Konstruktor pada StringBuffer dan fungsinya :

....
....
....
....
....
....
....
....

G. Lembar Kerja Siswa

13. Kegiatan 13 : Array (Deklarasi Array)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 13 ini siswa diharapkan dapat :

1) Memahami array sebagai media penyimpanan
2) Menyajikan array sebagai penyimpan data

B. Uraian Materi

1) Deklarasi Array

Array digunakan untuk membuat variabel bisa menampung beberapa data dengan tipe data yang sama alias satu tipe data. Ciri khas varibel yang menggunakan array adalah terdapat simbol **[ ].**

Array menggunakan indeks integer untuk menentukan urutan elemen-elemennya, dimana elemen pertamanya dimulai dari indeks 0,elemen kedua memiliki indeks 1, dan seterusnya.

Untuk mengetahui panjang array yang telah kita buat, kita dapat memakai property *length*, dengan format sebagai berikut :

`var_array.length` → total elemen array pada dimensi 1.

`var_array[i].length` → total elemen array pada dimensi 2 untuk indeks ke-i pada dimensi 1.

`var_array[i][i].length` → total elemen array pada dimensi 3 untuk indeks ke-I pada dimensi 1 dan indeks ke-I pada dimensi 2dan seterusnya.

Isi dari suatu array dapat kita copy pada array yang lain dengan memanfaatkan method arraycopy() pada class System. Format penulisannya sebagai berikut :

```
System.arraycopy (array1, p1, array2, p2, n);
```

Array memiliki dua tipe, yaituSingle Dimension dan Multi Dimension, sedangkan Proses pendeklarasian variabel yang menggunakan array ada 2 tipe :

- ✓ **Array Single Dimension**
  - **Pendeklarasian Variabel Menggunakan Array**

Contoh pendeklarasian Array 1 Dimensi :

Listing Program

```
1  public Array{
2  int [] array1;
3  int array2[];
4  double array3[];
5  char[] kata;
6  }
```

Pendeklarasian array diatas belum mempunyai nilai dan ukuran yang artinya masih kosong (empty). Sehingga array tersebut belum dapat dipakai atau diakses. Untuk mengalokasikan memory array dapat menggunakan variable new sehingga panjangnya diketahui.

Listing Program

```
1  public interface ArrayNew{
2  int [] array1;
3  array1 = new int[100];
4  //atau
5  int [] array2 = new int[100];
6  }
```

atau langsung mendeklarasikan item/nilai menjadi isi variable array secara langsung.

Listing Program

```
1  public interface ArrayNew{
2  Int [] array1={1,2,3,4,5,6,7,8,9};
3  //untuk tipe angka char huruf
   []={'a','b','c','d','e','f'};
4  //untuk huruf
5  }
```

Lantas bagaimana caranya agar kita mengetahui panjang arraynya, di atas variabel nilai menggunakan array langsung diinisalisasi jika ingin mengetahui panjang arraynya **nilai.length.**

Variabel yang menggunakan array, array ditentukan length/panjangnya terlebih dahulu, sehabis itu baru di inisialisasi.

Contoh :

Listing Program

```
1 public class TestArray{
2 public static void main (String args []) {
3 int nilai [] = new int [3];
4 nilai [0] = 70;
5 nilai [1] = 80;
6 nilai [2] = 65;
7 double ratarata = 0.0;
8 for (int i=0; i<nilai.length; i++)ratarata+=nilai [i];
9 ratarata/=nilai.length;
10 System.out.println("Nilai rata-rata = "+ ratarata);
11 }
12 }
```

Perlu diingat bahwa index array dimulai dari 0.

✓ **Array MultiDimension**

Pada konsep array multidimension, konsepnya sama seperti baris dan kolom pada tabel.

- **Pendeklarasian Variabel Menggunakan Array Tipe Pertama**

Sama seperti pada pendeklarasian variabel menggunakan array single dimension, perbedaannya kita harus menentukkan panjang kolomnya, jadi kita menentukan panjang baris dan panjang kolomnya.

Contoh :

Listing Program

```
1  import java.text.NumberFormat;
2  public class Array2{
3  public static void main (String args []){
4  NumberFormat nf = NumberFormat.getInstance();
5  nf.setMaximumFractionDigits(3);
6  int nilai [][]=new int [2][3];
7  nilai [0][0]=85;
8  nilai [0][1]=81;
9  nilai [0][2]=78;
10 nilai [1][0]=65;
11 nilai [1][1]=73;
12 nilai [1][2]=71;
13 String MK[]={"RPL","PBO"};
14 double aratarataMK[]= new double[nilai.length];
15 for (int i=0; i<nilai.length; i++){
16 for (int j=0; j<nilai[0].length; j++){
17 ratarataMK[i]+=nilai [i][j];
18 }
19 }
20 ratarataMK[i]/=nilai[0].length;
21 }
22 System.out.println("Nilai Mata Pelajaran\n");
23  System.out.println  ("MK\  tMinggu1  \  tMinggu2\
tMinggu3\ tRata-Rata");
24 for (int i=0; i<nilai.length; i++){
```

```
25 System.out.print(MK[i]+"\t");
26 for (int j=0;j<nilai[0].length; j++){
27 }
28 System.out.print(nf.format(ratarataMK[i]+"\n");
29 }
30 }
31 }
```

Seperti biasa indeks baris dimulai dari 0 dan indeks kolom dimulai dari 0.

- **Pendeklarasian Variabel Menggunakan Array Tipe Kedua**

Tipe kedua, proses pendeklarasian langsung dengan inisialisasinya :

Listing Program

```
char [][] abjad={{'A','B'},{'C','D'},{'E','F'}};
```

Untuk mengetahui panjang baris **abjad.length,** sedangkan untuk mengetahui panjang kolom bisa gunakan **abjad.length[0],** Kenapa harus 0 parameter untuk mengetahui panjang kolomnya?, alasannya karena sudah pasti indeks array dimulai dari 0.

- ✓ **Memproses Array**

Ketika memproses elemen-elemen array, seringkali kita menggunakan loop for, karena semua elemen suatu array bertipe sama, dan ukuran array yang telah diketahui, maka sangat cocok untuk menggunakan loop for.

Sebagai contoh :

Listing Program

```
1 double [] myList = new double [10];
```

Berikut ini adalah contoh pemrosesan array :

1. Loop berikut ini menginisialisasi array myList dengan nilai-nilai dari pengguna:

Listing Program

```
1  java.util.Scanner  masukkan  =  new  java.util.Scanner
(System.in);
2  System.out.print("masukkan  "  +myList.length+  "buah
nilai : ");
3 for (int i=0; i<myList.length;i++)
4 myList[i]=masukkan.nextDouble();
```

2. Loop berikut ini menginisialisasi array myList dengan nilai-nilai acak antara 0.0 sampai 100.0, tetapi kurang dari 100.0:

Listing Program

```
1 for (int i=0; i<myList.length; i++){
2 myList [i]= Math.random()*100;
```

3. Untuk menampilkan suatu array, kita harus menampilkan setiap elemen array menggunakan suatu loop sebagai berikut :

Listing Program

```
1 for (int i=0; i<myList.length; i++){
2 System.out.print (myList[i] +" ");
2  }
```

C. Rangkuman

Array merupakan bentukan yang menyediakan penyimpanan sejumlah item bertipe sama. Item – item array dapat berupa data sederhana atau komposit.

Array di java dideklarasikan dengan kurung siku [ ]. Di java kita harus menyatakan ukuran array secara eksplisit saat melakukan penciptaan array menggunakan operator *new ( )* atau dengan mendaftarkan item – item untuk array pada saat penciptaan.

Array terbagi menjadi array 1 dimensi, 2 dimensi, 3 dimensi dan multidimensi. Setiap array mempunyai indeks yang diawali dengan 0 sampai n data. Pemanggilan nilai / data melaui indeks tersebut.

D. Tugas

Tugas 1

Buatlah suatu array yang berisi 10 nama teman kamu satu kelas !

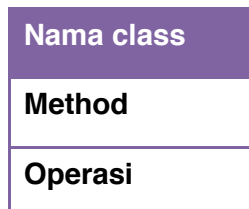
Lalu tampilkan nama temanmu melalui inputan sesuai indeks , tampilkan ke layar hasilnya

Tugas 2

Buatlah suatu array untuk melakukan pencarian indeks, isikan 10 data yang telah kamu tentukan sendiri.

Lalu tampilkan data melalui inputan sesuai indeks dan tampilkan ke layar hasilnya

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama classl yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
14.	
15.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa kegunaan array ?
2. Bagaimana anda mengakses elemen suatu array ?
3. Jelaskan perbedaan array 1 dimensi , array 2 dimensi dan array multi dimensi !
4. Apakah output dari kode berikut ini !

Listing Program

```
1  int x =30;
2  int[] angka= new int[x];
3  x = 60;
3  System.out.println("x adalah "+ x);
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Kegunaan array

....

....

....

....

....

....

LJ- 02 : Cara mengakses elemen suatu array

....

....

....

....

....

....

G. Lembar Kerja Siswa

14. Kegiatan 14 : Array (Collection Framework Interface)

1. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 14 ini siswa diharapkan dapat :

1. Memahami penggunaan collection
2. Menyajikan colecctioon dalam penyimpanan data

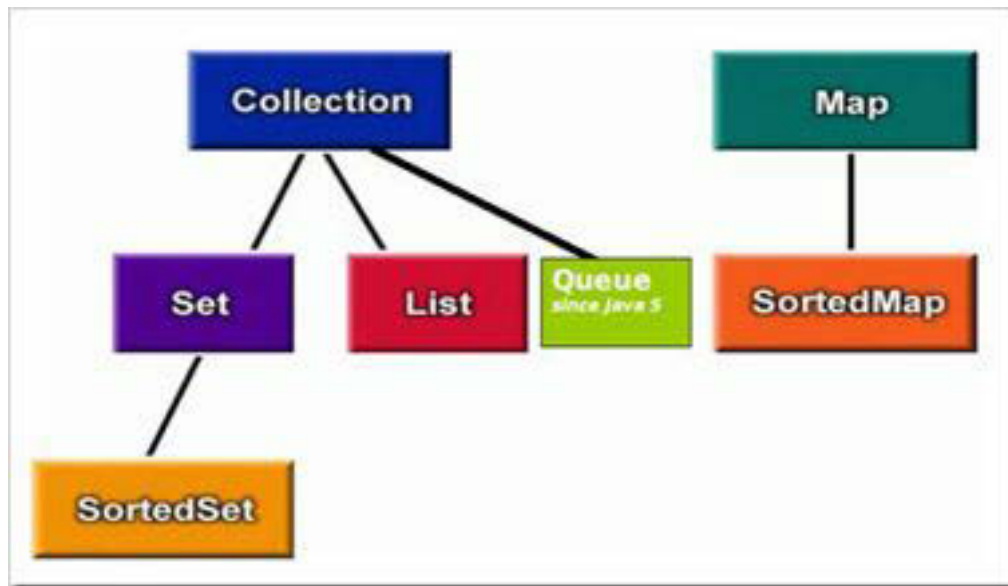
2. Uraian Materi

1) Collection Framework

Collection merupakan istilah umum yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan suatu teknik tertentu pula. Semua class yang berhubungan dengan pengelompokan objek ini dalam java tergabung dalam *Java Collection Framework*, dimana Framework ini diletakan dalam package java.util dan mempunyai dua interface utama, yaitu collection dan map.

Collection digunakan untuk menyimpan, mengambil, dan memanipulasi data-data. Salah satu collection paling sederhana adalah Array. Collection terbagi menjadi 3 kelompok yaitu Set, List dan Map. Berikut ini adalah struktur hierarki interface dan class yang termasuk dalam kelompok collection ini.

Beberapa kelebihan utama menggunakan collection framework antara lain:

- Mengurangi effort dalam membuat program, karena sudah tersedia struktur data dan algoritma tanpa harus menulis sendiri.
- Meningkatkan performa, karena setiap implementasi dapat berfungsi maksimal sesuai kasus yang ada.
- Mudah dipelajari, sehingga mengurangi effort untuk mempelajari cara menggunakan API.
- Dapat dikembangkan dan fleksibel terhadap tipe object yang ada dalam Collection.

Jenis Pengelompokan Collection ini merupakan pengelompokan satu dimensi. Berdasarkan teknik Collection Interface pengelompokannya terbagi menjadi tiga kelompok yaitu set, list, dan queue.

1. Set : Koleksi yang tidak boleh ada duplikasi nilai, dengan kata lain semua value dari class collection yang mengimplementasikan **interface Set** tidak boleh ada duplikasi nilai di dalamnya. Set merupakan turunan langsung dari **collection** class

2. List : Koleksi boleh ada duplikasi nilai dan terurut.

3. Map : Object yang memetakan object ke nilai.

4. Queue : Collection yang menjaga urutan elemenya berdasarkan FIFO.

5. SortedMap : sebuap Map yang memelihara elemen key-nya terurut secara ascending.

6. SortedSet: Sebuat set yang memelihara pemetaan elemennya secara ascennding.

Kemudian ada 5 implementasi dasar yang digunakan pula, yaitu Hash Table, Resizable Array, Balanced Tree, Linked List, dan Hash Table + Linked List.

		Implementations				
		Hash Table	Resizeable Array	Balanced Tree	Linked List	Hash Table + Linked List
Interfaces	Set	HashSet		TreeSet		LinkedHashSet
	List		ArrayList		LinkedList	
	Map	HashMap		TreeMap		LinkedHashMap

Java Collections Framework terbagi menjadi tiga kelompok:

- **Set**

Set mengikuti model himpunan, dimana objek/anggota yang tersimpan dalam Set harus unik. Urutan maupun letak dari anggota tidaklah penting, hanya keberadaan anggota saja yang penting. Class-class yang mengimplementasikan interface Set adalah HashSet. Interface SortedSet merupakan subInterface dari interface Set. Untuk mengurutkan Set, kita dapat menggunakan class yang mengimplementasikan interface SortedSet yaitu clas TreeSet.

- **List**

List digunakan untuk menyimpan sekumpulan objek berdasarkan urutan masuk (ordered) dan menerima duplikat. Cara penyimpanannya seperti array, oleh sebab itu memiliki posisi awal dan posisi akhir, menyisipkan objek pada posisi tertentu, mengakses dan menghapus isi list, dimana semua proses ini selalu didasarkan pada urutannya. Classclass yang mengimplementasikan interface List adalah Vector, Stack, Linked List dan Array List.

Terdapat interface Queue yang cara penyimpanan seperti List, interface ini menyimpan objek menggunakan metode FIFO (First In First Out) yaitu objek yang masuk pertama keluar pertama. Class-class yang mengimplementasikan interface Queue adalah PriorityQueue dan LinkedList. Data yang tersimpan pada objek

PriorityQueue akan diurutkan, data tersebut harus mengimplementasikan objek Comparable atau Comparator.

- **Map**

Perbedaaan mendasar map dengan collection yang lain, untuk menyimpan objek pada Map, perlu sepasang objek, yaitu key yang bersifat unik dan nilai yang disimpan. Untuk mengakses nilai tersebut maka kita perlu mengetahui key dari nilai tersebut. Map juga dikenal sebagai dictionary/kamus. Pada saat menggunakan kamus, perlu suatu kata yang digunakan untuk pencarian. Class-class yang mengimplementasikan Map adalah Hashtable,HashMap, LinkedHashMap. Untuk mengurutkan Map menggunakan interface SortedMap, class yang mengimplementasikan interface tersebut adalah TreeMap.

Contoh Program ArrayList :

Listing Program

```
1  import java.util.ArrayList;
2  public class array {
3  public static void main(String[] args) {
4  ArrayList<Human> list1 = new ArrayList<Human>();
5  for(int i = 0; i < 10; i++) {
6  list1.add(new Human("Human " + i));
7  }
8  for(int i = 0; i < list1.size(); i++) {
9  System.out.println(list1.get(i).name);
10 }
11 }
12 }
13 class Human {
14 public String name;
15 public Human(String name) {
16 this.name = name;
17 }
18 }
```

Contoh Program LinkedList :

Listing Program

```
1  import java.util.LinkedList;
2  public class LinkListCollection {
3  public static void main(String[] args) {
4  LinkedList<String> A = new LinkedList<String>();
5  String[] nama = {"David","Alfa","Benny"};
6  //Tambah data data diambil dari array nama;
7  for(int nList = 0;nList<nama.length;nList++){
8  A.add(nama[nList]);
9  }
10 //Tampil Data
11 System.out.println("Data Asli : ");
12 for(int nList = 0;nList<nama.length;nList++){
13  System.out.println  ("Indeks  "+nList+"  :  "+A.get
(nList));
14 }
15 System.out.println("\nTambah data di Index ke-3 : ");
16 A.add(3, "Danni");
17 for(int nList = 0;nList<=nama.length;nList++){
18  System.out.println("Indeks"  +nList+  ":  "  +A.get
(nList));
19 }
20 System.out.println("\nDelete data di Index ke-2 : ");
30 A.remove(2);
31 for(int nList = 0;nList<nama.length;nList++){
32  System.out.println("Indeks  "  +nList+  ":  "  +A.get
(nList));
33 }
34 System.out.println("\nTambah data di Awal list : ");
35 A.addFirst("Marcelo");
36 for(int nList = 0;nList<=nama.length;nList++){
37  System.out.println("Indeks  "  +nList+  ":  "  +A.get
(nList));
```

```
38 }
39 System.out.println("\nTambah data di Akhir list : ");
40 A.addLast("Ferdi");
41 for(int nList = 0;nList<=nama.length+1;nList++){
42 System.out.println("Indeks"+nList+": " +A.get (nList));
43 }
44 System.out.println("\nTambah data di Akhir list : ");
45 A.remove(4);
46 for(int nList = 0;nList<=nama.length;nList++){
47  System.out.println("Indeks  "  +nList+  ":  "  +A.get
(nList));
48 }
49 }
50 }
```

Contoh Program HashSet :

1. Menggunakan method hashCode dari object yang akan dimasukkan ke dalam elemen. Sehingga semakin efisien method hashCode dari elemen yang dimasukkan, semakin baik performa aksesnya.
2. HashSet tidak ordered. Sehingga ketika kita meng-iterate HashSet, urutanya tidak dapat diprediksi.

Listing Program

```
1  import java.util.ArrayList;
2  import java.util.HashSet;
3  public class HashSetDemo {
4  public static void main(String[] args) {
5  HashSet<Human> set = new HashSet<Human>();
6  ArrayList<Human> list = new ArrayList<Human>();
7  Human a = new Human("A");
8  Human b = new Human("B");
9  Human c = new Human("A");
10 set.add(a);
11 set.add(b);
12 set.add(c);
```

```
13 list.add(a);
14 list.add(b);
15 list.add(c);
16 System.out.println("Print Set");
17 for(Human h : set) {
18 System.out.println(h.name);
19 }
20 System.out.println("Print List");
21 for(Human h : list) {
22 System.out.println(h.name);
23 }
24 }
25 }
26 class Human {
27 public String name;
28 public Human(String name) {
29 this.name = name;
30 }
31 @Override
32 public boolean equals(Object obj) {
33 if(!(obj instanceof Human)) {
34 return super.equals(obj);
35 } else {
36 Human comp = (Human)obj;
37 return comp.hashCode() == obj.hashCode();
38 }
39 }
40 @Override
41 public int hashCode() {
42 int hash = 7;
43 hash = 53 * hash + (this.name!=null
   ? this.name.hashCode() :0);
44 return hash;
45 }
```

```
46 }
```

Contoh Program HashMap :

1. Semakin efisien mehtod hashCode dari key, maka semakin baik performance yang akan kita dapatkan.
2. HashMap mengijinkan sebuah null key dan multiple null value pada collection.

Listing Program

```
1  import java.util.HashMap;
2  import java.util.Iterator;

3  public class HashMapDemo {
4  public static void main(String[] args) {
5  HashMap<String, String> map = new HashMap<String,
   String>();
6  map.put("rumah", "Tipe 45");
7  map.put("mobil", "BMW");
8  map.put("bunga", "Mawar");
9  System.out.println("Sebelum diubah");
10 Iterator<String> ite = map.keySet().iterator();
11 while(ite.hasNext()) {
12 System.out.println(map.get(ite.next()));
13 }
14 System.out.println("Setelah diubah");
15 map.put("mobil", "Ferrari");
16 for(String key : map.keySet()) {
17 System.out.println(map.get(key));
18 }
19 }
20 }
```

Contoh Program Vektor :

Listing Program

```
1   import java.util.Vector;
2   Public class VectorExample {
3   public static void main(String[] args) {
4   Vector<String> vc=new Vector<String>();
5   //<E> Element type of Vector e.g. String, Integer,
Object
6   // add vector elements
7   vc.add("Vector Object 1");
8   vc.add("Vector Object 2");
9   vc.add("Vector Object 3");
10  vc.add("Vector Object 4");
11  vc.add("Vector Object 5");
12  // add vector element at index vc.add(3,"Element at
fix position");
13  //vc.size() inform number of elements in Vector
14  System.out.println("Vector Size :"+vc.size());
15  //get elements of Vector for(int i=0;i<vc.size();i++)
16  {
17  System.out.println("Vector Element"+i+":"+vc.get(i));
18  }
19  }
20  }
```

Contoh Program Iterator :

Listing Program

```
1   import java.util.*;
2   class IteratorDemo {
3   public static void main(String args[]) {
```

```
4  // create an array list
5  ArrayList al = new ArrayList();
6  // add elements to the array list
7  al.add("C");
8  al.add("A");
9  al.add("E");
10 al.add("B");
11  al.add("D");
12  al.add("F");
13 // use iterator to display contents of al
14 System.out.print("Original contents of al: ");
15 Iterator itr = al.iterator();         while
    (itr.hasNext()) {
16 Object element = itr.next();
17 System.out.print(element + " ");
18 }
19 System.out.println();
20 // modify objects being iterated
21 ListIterator litr = al.listIterator();         while
    (litr.hasNext()) {
22 Object element = litr.next();
23 litr.set(element + "+");
24 }
25 System.out.print("Modified contents of al: ");
26 itr = al.iterator();
27 while (itr.hasNext()) {
28 Object element = itr.next();
29 System.out.print(element + " ");
30  }
31 System.out.println();
32 // now, display the list backwards
33 System.out.print("Modified list backwards: ");
34 while (litr.hasPrevious()) {
35 Object element = litr.previous();
```

```
36 System.out.print(element + " ");
37 }
38 System.out.println();
39 }
40 }
```

Contoh Program Stack :

Listing Program

```
1  import java.io.*;
2  import java.util.*;
3  public class StackImplement {
4  Stack<Integer> stack = new Stack<Integer>();
5  String str;
6  int num, n;
7  public static void main(String[] args) {
8  StackImplement q = new StackImplement();
9  }
10 StackImplement() {
11 try {
12 BufferedReader bf = new BufferedReader(
13 new InputStreamReader(System.in));
14 System.out.print("Banyak Data : ");
15 str = bf.readLine();
16 num = Integer.parseInt(str);
17 for (int i = 1; i <= num; i++) {
18 System.out.print("Masukan Elemen " + i + " : ");
19 str = bf.readLine();
20 n = Integer.parseInt(str);
21 stack.push(n);
22 }
23 } catch (IOException e) {
24 }
25 System.out.println("Stack : ");
```

```
26 while (!stack.empty()) {
27 System.out.print(stack.pop() + "  ");
28 }
29 System.out.println();
30 }
31 }
```

Contoh Program Queue :

Queue merupakan model pengelompokan berdasarkan metode antrian suatu prioritas tertentu(contoh FIFO-First In First Out). Beberapa Class java yang mengimplementasi interface *Queue* ini antara lain *PriorityQueue* dan *LinkedList*.Dalam queue terdapat fungsi enqueue yang digunakan untuk mengatur inputan data yang akan masuk antrian, pada fungsi ini juga diperiksa apakah antrian sudah penuh atau belum, jika sudah penuh maka tidak dapat diisi lagi. Fungsi dequeue digunakan untuk mengatur data agarkeluar dari antrian secara tertib, karena queue menggunakan prinsip FIFO.

Contoh :

Listing Program

```
1    import java.io.*;
2    import java.util.*;
3    public class QueueJava{
4    String str;
5    Int num;
6   public static void main(String[] args){
7    QueueJava q = new QueueJava();
8    }
9    public QueueJava(){
10   try{
11  LinkedList<Integer> list = new LinkedList<Integer>();
12   BufferedReader bf = new BufferedReader(
13   new InputStreamReader(System.in));
14   System.out.print("Banyak Data : ");
15   str = bf.readLine();
16   if((num = Integer.parseInt(str)) == 0){
17   System.out.println("Anda menekan angka nol.");
```

```
18  System.exit(0);
19  }
20  else{
21  for(int i = 0; i < num; i++){
22  System.out.print("Masukan Elemen "+i+" : ");
23  str = bf.readLine();
24  int n = Integer.parseInt(str);
25  list.add(n);
26  }
27  }
28  System.out.println("\nElement Pertama    : "
29  + list.removeFirst());
30  System.out.println("Element Terakhir  : "
31  + list.removeLast());
32  System.out.println("Element Tengah    : ");
33  while(!list.isEmpty()){
34  System.out.print(list.remove() + " ");
35  }
36  System.out.println("");
37  }
38  catch(Exception e){
39  System.out.println(e.getMessage()
40  + " adalah String.");
41  System.exit(0);
42  }
43  }
44  }
```

3. Rangkuman

Collection merupakan istilah yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan teknik tertentu.Semua class yang berhubungan dengan pengelompokkan objek ini tergabung dalam Java Collection Framework.

D. Tugas

Tugas 1

Buatlah suatu program menampilkan nilai berikut

a={1,2,3,4,5,}

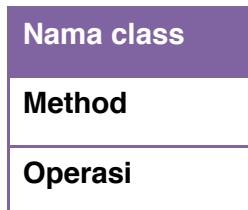
b={5,6,7,8,9,10}

cari

a. a U b (union)

b. a n b (intersect)

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama kelas yang akan digunakan.
2. Tentukan variabel yang akan digunakan
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
14.	
15.	
16.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Menurut anda apa yang dimaksud dengan collection ? Jelaskan !
2. Jelaskan fungsi list, set dan map !
3. Apa perbedaan stack dan queue ?
4. Masukkan sepotong method ini ke kelas main.
   Analisislah method apa tersebut ? Tulislah hasil outputnya !

Listing Program

```
1 String[]
2 electronics = {"Computer", "Laptop", "iPhone",
"iPad"};
3 List list = new Array (Arrays.asList
(electronics));
4 System.out.println("ArrayList of Electronics
:" + list);
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi Collection

....

....

....

....

....

....

G. Lembar Kerja Siswa

15. Kegiatan 15 : Array (Set dan Sorting)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 15 ini siswa diharapkan dapat :

1) Memahami fungsi Set sebagai media penyimpanan
2) Menyajikan fungsi Set sebagai penyimpan data

B. Uraian Materi

1) Set

Himpunan (set) adalah kumpulan Object yang mana tidak boleh ada dua dari objek yang sama di dalam satu himpunan. Objek obj1 dan obj2 adalah objek yang sama jika obj1.equals(obj2) menghasilkan nilai true

Set mengimplementasikan metode umum pada Collection dengan sedikit modifikasi sehingga tidak memiliki objek yang sama di dalamnya. Misalnya, jika set adalah objek bertipe Set maka set.add(obj) tidak melakukan apa-apa jika obj sudah ada di dalam set.

Java memiliki dua kelas yang mengimplementasikan interface Set, yaitu java.util.TreeSet dan java.util.HashSet.

Selain sebagai Set, objek bertipe TreeSet memiliki sifat di mana elemen-elemennya terurut dalam urutan menaik. Iterator dari suatu TreeSet akan mengunjungi elemen dalam set tersebut dalam urutan menaik.

TreeSet tidak bisa menyimpan semua objek, karena objek yang disimpan harus bisa diurutkan. Atau dengan kata iain, objek yang disimpan dalam TreeSet harus mengimplementasikan interface Comparable dan obj1.compareTo(obj2) harus bisa membandingkan 2 objek obj1 dan obj2 dengan baik. Atau cara lainnya adalah dengan menggunakan Comparator yang bisa dimasukkan sebagai parameter dalam konstruktor ketika TreeSet dibuat. Di sini Comparator digunakan untuk membandingkan objek yang akan dimasukkan di dalam set.

Pada implementasi TreeSet, elemen di dalamnya disimpan dalam bentuk yang mirip dengan pohon pengurutan biner. Pohon yang digunakan bersifat seimbang (balanced) yang artinya semua simpul daunnya berjarak sama dari akar pohonnya. Dengan menggunakan pohon seimbang, kita bisa memastikan bahwa kita bisa mencapai semua simpul daunnya dengan secepat mungkin. Menambah dan mengurangi simpulnya juga bisa dilakukan dengan cepat.

Karena TreeSet mengurutkan elemen-elemennya dan menjamin tidak ada duplikasi objek, TreeSet sangat berguna untuk beberapa aplikasi. Misalnya, kita akan membuat program untuk menghitung semua kata dalam suatu artikel dan menampilkan semua kata yang ditemukan. List kata-kata dalam artikel itu harus diurutkan, kemudian kata-kata yang sama dihapus.

Contoh syntax TreeSet :

Listing Program

```
1 TreeSet kata = new TreeSet();
2 while masih ada data dari input :
3 kt = kata berikutnya dari input
4 kata.add(kt);
5 Iterator iter = kata.iterator();
6 while (iter.hasNext()):
7 Tampikan iter.next() ke layar.
```

Perintah baris kedua di atas menambah semua elemen dari kol ke dalam set. Karena berbentuk Set, maka duplikasi akan dihapus. Karena berbentuk TreeSet maka semua elemennya akan terurut. Jika kita ingin data tersebut disimpan dalam struktur data lain, kita bisa mengubahnya dengan mudah. Misalnya, untuk membuat ArrayList dari TreeSet, bisa kita tulis dengan

Listing Program

```
1  TreeSet set = new TreeSet();
2  set.addAll(kol);
3  ArrayList list = new ArrayList();
```

```
4  list.addAll(set);
```

2) Sorting

Sorting adalah proses menyusun elemen – elemen dengan tata urut tertentu dan proses tersebut terimplementasi dalam bermacam aplikasi. Kita ambil contoh pada aplikasi perbankan. Aplikasi tersebut mampu menampilkan daftar account yang aktif. Hampir seluruh pengguna pada sistem akan memilih tampilan daftar berurutan secara ascending demi kenyamanan dalam penelusuran data. Dalam artian sorting digunakan untuk mengurutkan sesuatu (misalnya : kata, buku telepon , dll).

Sorting yang kita terapkan menggunakan tipe data array agar pemahaman serta pengimplementasiannya lebih mudah. Pada umumnya terdapat dua jenis pengurutan :

- Ascending (Naik). Terurut Ascending : 2 3 8 10 15 22
- Descending (Turun). Terurut Descending : 22 15 10 8 3 2

Macam Sorting yaitu :

✓ **Buble Sort**

Bubble sort adalah metode sorting termudah. Diberikan nama “bubble” karena konsep dari algoritmanya diibaratkan seperti gelembung air untuk elemen struktur data yang seharusnya pada posisi awal. Bubble sort mengurut data dengan cara membandingkan elemen sekarang dengan elemen berikutnya. Dimana cara kerjanya adalah dengan berulang-ulang melakukan proses looping (perulangan) terhadap elemen-elemen struktur data yang belum diurutkan. Nilai dari masing-masing elemen akan dibandingkan selama proses looping tersebut .jika selama proses looping tersebut ditemukan ada urutannya tidak sesuai dengan permintaan, maka akan dilakukan proses penukaran (swap). Secara tidak langsung, algoritma dari program ini seolah-olah menggeser satu demi satu elemen dari kanan ke kiri atau dari kiri ke kanan tergantung pada jenis pengurutannya. Jenis pengurutan sorting ada 2 yaitu asscending dan descending. Dimana asscending itu mengurut data dari kecil ke besar dan descending itu mengurut data dari besar ke kecil. Jika semua elemen sudah diperiksa oleh fungsi bubble sort, dan tidak ada pertukaran lagi atau semua nilai

sudah sesuai, maka saat itu program bubble sort akan berhenti bekerja. Pola Bubble sort yaitu bertukar dengan data sebelahnya.

✓ **Insertion Sort**

Algoritma insertion sort pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan (meja pertama), dan yang telah diurutkan (meja kedua). Elemen pertama yang diambil dari bagian array yang belum diurutkan dan kemudian diletakkan pada posisinya sesuai dengan bagian lain dari array yang telah diurutkan. langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan. Pola dari Insertion Sort menggurutkan n atau 2 elemen dari terdepan dst.

✓ **Selection Sort**

Merupakan Kombinasi antara sorting dan searching. Metode selection sort merupakan perbaikan dari metode bubble sort dengan mengurangi jumlah perbandingan. Selection sort merupakan metode pengurutan dengan mencari nilai data terkecil dimulai dari data diposisi 0 hingga diposisi N-1. Jika terdapat N data dan data terkoleksi dari urutan 0 sampai dengan N-1. Selama proses, perbandingan dan pengubahan, hanya dilakukan pada indeks permbandingnya saja, pertukaran data secara fisik terjadi pada akhir proses.sesuai dengan itu maka algoritma pengurutan data secara Selection adalah sebagai berikut :

1. Cari data terkecil dalam interval j = 0 sampai dengan j = N-1
2. Jika pada posisi pos ditemukan data yang terkecil, maka tukarkan data diposisi pos dengan data di posisi i jika k.
3. Ulangi langkah 1

✓ **Merge Sort**

Pengurutan algoritma Merge Sort membuat pengurutan dengan membagi 2 dan menggabungkannya. Metoda ini cukup efisien untuk diterapkan. Sama dengan Quick Sort, algoritma Merge Sort adalah dasar pembagian dan penyelesaiannya. Pertama urutan atau elemen data awal diurutkan dengan membaginya menjadi 2 bagian (Devide). Setengahnya diurutkan dengan bebas (Conquer). Kemudian 2 bagian itu digabungkan dengan cara diurut sesuai dengan urutan (Combine).

1. Devide, yakni memilih masalah menjadi sub-masalah.
2. Conquer, yakni menyelesaikan sub-masalah tersebut secara rekursi.
3. Kombinasi/Penggabungan, menggabungkan solusi dari sub-masalah.

✓ **Quick Sort**

Pengertian Quick Sort adalah algoritma yang dijalankan sebagai akibat dari terlalu banyaknyadaftar yang diurutkan, dengan menghasilkan lebih banyak daftar yang diurutkan sebagai output. Algoritma merge ini disesuaikan untuk mesin drive tape. Penggunaannya dalam akses memori acak besar yang terkait telah menurun, karena banyak aplikasi algoritma merge yang mempunyai alternatif lebih cepat ketika kamu memiliki akses memori acak yang menjaga semua data. Hal ini disebabkan algoritma ini membutuhkan setidaknya ruang atau memori dua kali lebih besar karena dilakukan secara rekursif dan memakai dua tabel.

Contoh Program Sorting :

Listing Program

```
1  public static void main (String []riris){
2  int menu=0;
3  do{
4  try{
5  menu=Integer.parseInt(JOptionPane.showInputDialog("Pilih
   Menu :"+ "\n1.Bubble Sort"+"\n2.Selection
   Sort"+"\n3.Insertion Sort"+"\n4.Exit"));
6  switch(menu){
7  case 1: int n= Integer.parseInt
   (JOptionPane.showInputDialog("Banyaknya data?"));
8  int data []  = new int [n];
9  System.out.println("Sorting menggunakan Bubble Sort");
10 System.out.println("Data sebelum diurutkan :");
11 for(int a=0; a<n; a++){
12 data[a]=Integer.parseInt
   (JOptionPane.showInputDialog("Data ke-" + (a+1)));
13 System.out.println("Data ke-" +a + "=" + data [a] );
14 }
```

```
15 int temp;
16 for (int i=1;i<n;i++){
17 for(int j=n-1; j>=i; j--){
18 if (data[j]<data[j-1]){
19 temp=data[j];
20 data[j]=data[j-1];
21 data[j-1]=temp;
22 }
23 }
24 }
25 System.out.println("Data yang sudah diurutkan: ");
26 for (int i=0;i<data.length;i++)
27 System.out.println("Data ke-"+i+ "=" +data [i]);
28 System.out.println("------------------------------
   ");break;
29 case 2: int m= Integer.parseInt
   (JOptionPane.showInputDialog("Banyaknya data?"));
30 int nilai []= new int [m];
31 int index,large;
32 System.out.println("Sorting menggunakan Selection Sort");
33 System.out.println("Data Sebelum Diurutkan");
34 for(int a=0; a<m; a++){
35 nilai
   [a]=Integer.parseInt(JOptionPane.showInputDialog("Data
   ke-"+(a+1)));
36 System.out.println("Data ke-"+a+"="+nilai[a]);
37 }
38 for (int i=m-1; i>0; i--){
39 large=nilai[0];
40 index=0;
41 for (int j=1; j<=i; j++){
42 if (nilai [j]>large){
43 large=nilai [j];
44 index=j;
```

```
45 }
46 }
47 nilai[index]=nilai [i];
48 nilai[i]=large;
49 }
50 System.out.println("Data yang sudah diurutkan : ");
51 for (int i=0;i<m;i++)
52 System.out.println(nilai[i]+ " ");
53 System.out.println("------------------------------
   ");break;
54 case 3:
55 int l=
   Integer.parseInt(JOptionPane.showInputDialog("Banyaknya
   data?"));
56 int nilai2 []=new int [l];
57 int y;
58 System.out.println("Sorting menggunakan Insertion Sort");
59 System.out.println("Data Sebelum Diurutkan");
60 for(int a=0; a<l; a++){
61 nilai2[a]=Integer.parseInt(JOptionPane.showInputDialog("D
   ata ke-" + (a+1)));
62 System.out.println("Data ke-" +a + "=" + nilai2 [a] );}
63 System.out.println("Data Yang Sudah Diurutkan");
64 for (int j=1; j<l; j++){
65 y=nilai2[j];
66 for (int i=j-1; i>=0 && y<nilai2[i]; i--){
67 nilai2 [i+1]=nilai2 [i];
68 nilai2 [i]=y;
69 }
70 for (int i=0; i<l; i++)
71 System.out.print(nilai2[i]+ " ");
72 System.out.println("");
73 }
74 System.out.println("Arigato Gozaimasu :)");break;
75 case 4:
```

```
76 System.exit(0); break ;
77 default: JOptionPane.showMessageDialog(null, "pilihan
   antara 1-3"); break;
78 }
79 }catch(Exception e)
80 {
81 JOptionPane.showMessageDialog(null,"MASUKAN ANGKA", "anda
   salah", JOptionPane.ERROR_MESSAGE);
82 }
83 }while (menu!=4);
84 }
85 }
```

C. Rangkuman

Set merupakan pengelompokkan mengikuti model himpunan dimana setiap anggotanya harus unik. Sorting bisa didefinisikan sebagai suatu pengurutan data yang sebelumnya disusun secara acak, sehingga menjadi tersusun secara teratur menurut aturan tertentu. sorting yang kita terapkan menggunakan tipe data array agar pemahaman serta pengimplementasiannya lebih mudah. Pada umumnya metode yang digunakan untuk sorting adalah Buble\Exchange sort, Selection sort, Shell Sort, Quick sort.

D. Tugas

Tugas 1

Buatlah program mengurutkan data dengan menggunakan algoritma merge sort, gunakan jenis pengurutan ascending!

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa kegunaan set dan sorting ?
2. Sebut dan jelaskan macam sorting !
3. Mana Algoritma sorting yang menurut anda paling efektif ? Jelaskan alasan anda !
4. Analisis hasil dari source code ini & tentukan jenis algoritma apa !

Listing Program

```
1  public class Sort {
2  public static void main(String args[]){
3  int[] data={21,13,36,12,18,9,59,24};
4  int temp;
5  for (int i=1;i<data.length;i++){
6  for (int j=data.length-1;j>=i;j-){
7  if (data[j]<data[j-1]){
8  temp=data[j];
9  data[j]=data[j-1];
10 data[j-1]=temp;
11 }
12 }
13 }
14 for (int i=0;i<data.length;i++)
15 System.out.print(data[i]+" ");
16 }
```

```
17 }
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Kegunaan set dan sorting :

a) Set

....

....

....

....

b) Sorting

....

....

....

....

LJ- 02 : Sorting dan jenisnya

....

....

....

....

....

LJ- 03 : Algoritma paling efektif

....

....

....

....

....

....

16. Kegiatan 16 : Operasi File (Sistem File)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 16 ini siswa diharapkan dapat :

1) Menerapkan operasi file dan Input Output (I/O)
2) Menyajikan operasi file dan operasi Input

B. Uraian Materi

1) Sistem File

Data dan program pada memori komputer hanya bisa bertahan selama komputer itu nyala. Untuk tempat penyimpanan yang lebih lama, komputer menggunakan file, yaitu kumpulan data yang disimpan dalam hard disk, disket atau CD-ROM, USB stick, dan lain-lain. File disusun dalam direktori (atau sering juga disebut folder). Direktori bisa terdiri dari direktori lain atau file lain. Nama direktori dan file digunakan untuk mencari suatu file dalam komputer.

Program dapat membaca data dari file yang sudah ada. Program juga bisa membuat file baru atau menulis data ke dalam file yang sudah ada. Dalam Java, input dan output seperti ini bisa menggunakan aliran (stream). Data karakter yang bisa dibaca manusial dapat dibaca dari file dengan menggunakan objek dari kelas FileReader yang merupakan kelas turunan Reader. Data bisa ditulis dalam bentuk yang bisa dibaca manusia dengan menggunakan FileWriter yang merupakan kelas turunan dari Writer.

Untuk membaca atau menyimpan suatu file dalam format mesin, kelas I/O-nya adalah FileInputStream danFileOutputStream. Semua kelas ini didefinisikan dalam paket java.io.

Kelas File merepresentasikan suatu file ataupun direktori dalam sistem file melalui pathname.

Pathname :

- Unix: /usr/java/bin/javac
- Windows: c:\java\bin\javac

Path representation :

- UNIX: /bin:/usr/bin:/usr/local/bin
- Windows: c:\bin;c:\java\bin

Field, constructor dan method kelas File dapat dilihat pada dokumen Java API pada bagian java.io.File

Perlu dicatat bahwa applet yang didownload dari suatu jaringan pada umumnya tidak bisa mengakses file karena pertimbangan keamanan. Kita bisa mendownload dan menjalankan applet, yaitu dengan mengunjungi halaman web pada browser kita. Jika applet tersebut bisa digunakan untuk mengakses file pada komputer kita, maka orang bisa membuat applet untuk menghapus semua file dalam komputer yang mendownloadnya.

Untuk mencegah hal seperti itu, ada beberapa hal di mana applet yang didownload tidak bisa lakukan. Mengakses file adalah salah satu hal yang dilarang. Akan tetapi program desktop bisa memiliki akses ke file kita seperti program-program lainnya. Program desktop bisa melakukan akses file yang dijelaskan pada bagian ini.

2) I/O

Program komputer bisa berguna jika ia bisa berinteraksi dengan dunia lain. Interaksi di sini maksudnya input/output atau I/O. Pada bab ini, kita akan melihat input output pada file dan koneksi jaringan (network). Pada Java, input/output pada file dan jaringan dilakukan berdasarkan aliran (stream), di mana semua objek dapat melakukan perintah I/O yang sama. Standar output (System.out) dan standar input (System.in) adalah contoh aliran.

Banyak subrutin yang digunakan untuk bekerja dengan I/O melemparkan pengecualian yang wajib ditangani. Artinya subrutin tersebut harus dipanggil di dalam pernyataan try ... catch sehingga pengecualian yang terjadi bisa ditangani dengan baik.

Bahasa pemrograman Java membuat proses I/O menjadi lebih sederhana. Maksudnya seperti ini, untuk semua proses I/O Anda hanya memerlukan satu class untuk proses input (dan sumber input bisa dari mana saja) dan satu class

untuk proses output (dan tujuan output, juga. bisa ke mana saja). Semua class yang diperlukan untuk proses I/O ada di dalam paket java.io

C. Rangkuman

Class dasar I/O (input output) terdiri dari Reader, Writer, InputStream dan OutputStream hanya menyediakan operasi I/O sangat dasar. Misalnya, class InputStream memiliki metode instansi publicint read() throws IOException untuk membaca satu byte data dari aliran input. Jika sampai pada akhir dari aliran input metode read() akan mengembalikan nilai -1. Jika ada kesalahan yang terjadi pada saat pengambilan input, maka pengecualian IOException akan dilemparkan.

D. Tugas

Tugas 1

Buatlah program untuk menampilkan suatu inputan berupa jumlah barang, sehingga inputan selanjutnya adalah berdasar pada jumlah barang tersebut, misalnya:

```
Masukkan jumlah barang : 2
Harga barang ke 1 :1000
Harga barang ke 2 : 2000
====================
Total pembelian   : 3000
```

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan.

Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa fungsi kelas IO ?
2. Sebutkan dan jelaskan kategori kelas IO !
3. Analisis kelas dasar IO Reader, Writer, InputStream, dan OutputStream ?
4. Analisis dan tuliskan hasil dari source code dibawah ini

Listing Program

```
1  public class file {
2  public static void main(String args[]){
3  System.out.println(File.separator+"-
   "+file.separatorChar+"-
   "+File.pathSeparator+"-
   "+File.pathSeparatorChar);
4  }
5  }
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Fungsi Kelas IO :

....

....

....

....

....

....

LJ- 02 : Kategori kelas IO :

....

G. Lembar Kerja Siswa

17. Kegiatan 17 : Operasi File (FileInputStream dan FileOutputStream)

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 17 ini siswa diharapkan dapat :

1) Memahami FileInputStream dan FileOutputStream
2) Menerapkannya pada suatu program

B. Uraian Materi

1) FileInputStream dan FileOutputStream

Kedua class FileInputStream dan FileOutputStream memungkinkan operasi file stream-based. Kedua class ini bekerja sama dengan class File dan FileDescriptor.

- **Class File**

Class file merupakan representasi dari file dan direktori (path). Constructornya:

Listing Program

```
1   File (String pathname)
2   File(String parent, String child)
3   File(File Parent, String child)
4   File(URI uri)
```

Bentuk pertama membuat file dalam direktori tertentu dan dengan nama file tertentu. Bentuk kedua dan ketiga membuat file dengan posisi relatif terhadap direktori parent. Direktori relatif parent dapat berupa string ataupun direktori dari sebuah file. Child menyatakan sub-direktori atau nama file yang terletak relatif terhadap parent.

Sedangkan uri merupakan object URI (Uniform Resource Identifier). Misalnya, "file:/// D:\JAVA\Applet1.HTML". Bentuk URI merupakan spesifik platform.

Beberapa metode dari class ini adalah:

Metode	Keterangan
delete()	Menghapus file atau direktori
getName()	Mengambil nama file
getPath()	Mengambil nama path
getAbsolutePath()	Mengambila nama path absolut
getParent()	Mengambil direktori parent dari file
exists()	Apakah file ada?
canWrite()	Jika file writeable?
canRead()	Jika file readable?
isFile()	Apakah file valid?
isDirectory()	Apakah direktori valid?
isAbsolute()	Apakah nama file tidak relative?
lastModified()	Mengambil waktu terakhir file dimodifikasi
length()	Mengambil panjang file
mkdir()	Membuat direktori
renameTo()	Mengganti nama file
mkdirs()	Membuat direktori-tree
list()	Mengambil daftar file yang ada direktori

- **FileDescriptor**

Class ini tidak boleh kita buat objeknya.ia dipakai untuk menunjukkan descriptor dari file yang aktif. Isinya spesifik mesin, sehingga ia bersifat opaque dan strukturnya tidak terlihat.

- **FileInputStream**

Class ini memungkinkan file dibaca sebagai input dalam bentuk stream. Contructornya:

Listing Program

```
1  FileInputStream(Filefile)throws
2  FileNotFoundException
3  FileInputStream(FileDescriptor fdObj)
```

File akan dibuka untuk dibaca. Sebagai turunan dari InputStream maka class ini menurunkan beberapa metode darinya. Seperti read(), skip(), dan close(). Sebagai tambahan, class ini memiliki metode getFD() yang dipakai untuk mengambil object FileDescriptor.

Listing Program

```
1  import java.io.File;
2  import java.io.FileInputStream;
3  import java.io.IOException;
4  public class modul {
5  public static void main(String[] args) {
6  boolean areFilesIdentical = true;
7  File file1 = new File("D:\\file1.txt");
8  File file2 = new File("D:\\file2.txt");
9  if (!file1.exists() || !file2.exists()) {
10 System.out.println("File tidak ditemukan");
11 System.out.println(false);
12 }
13 System.out.println("length:" + file1.length());
14 if (file1.length() != file2.length()) {
15 System.out.println("panjang file 1 dan file 2 tidak
   sama");
16 System.out.println(false);
17 }
18 try {
19 FileInputStream fis1 = new FileInputStream(file1);
20 FileInputStream fis2 = new FileInputStream(file2);
21 int i1 = fis1.read();
22 int i2 = fis2.read();
```

```
23 while (i1 != -1) {
24 if (i1 != i2) {
25 areFilesIdentical = false;
26 break;
27 }
28 i1 = fis1.read();
29 i2 = fis2.read();
30 }
31 fis1.close();
32 fis2.close();
33 } catch (IOException e) {
34 System.out.println("IO exception");
35 areFilesIdentical = false;
36 }
37 System.out.println(areFilesIdentical);
38 }
39 }
```

- **FileOutputStream**

Class ini memungkinkan file ditulisi sebagai output dalam bentuk stream. Constructornya:

Listing Program

```
1  FileOutputStream(File file) throws
   FileNotFoundException
2  FileOutputStream(File file, Boolean append) throws
   FileNotFoundExcption
3  FileOutputStream(FileDescriptor fdObj)
```

File akan dibuka untuk ditulisi. Jika append sama dengan true maka data akan ditambahkan ke akhir file. Sebagai turunan dari OutputStream maka class ini menurunkan beberapa metode darinya. Seperti write() dan close(). Sebagai tambahan, class ini memiliki metode getFD() yang dipakai untuk mengambil object FileDescriptor.

Listing Program

```
1  import java.io.FileOutputStream;
2  public class modul {
3  public static void main(String[] args) {
4  try{
5  FileOutputStream                    fos=newFileOutputStream
   ("D:\\java.txt");
6  String teks="saya belajar FileOutputStream";
7  byte[]isi=teks.getBytes();
8  fos.write(isi);
9  fos.close();
10 }
11 catch(Exception e){
12 System.err.println("Error tulis ke file");
13 }
14 }
15 }
```

C. Rangkuman

Kedua class FileInputStream dan FileOutputStrean memungkinkan operasi file stream-based. Class file merupakan representasi dari file dan direktori (path). Beberapa metode dari class file adalah delete(), getName(), getPath(), getAbsolutePath(), dan masih banyak lagi.

Class FileDescriptor digunakan untuk menunjukkan descriptor dari file yang aktif. Class FileDescriptor bersifat opaque dan strukturnya tidak terlihat. Class fileInputStream memungkinkan file dibaca sebagai input dalam bentuk stream. Sebagai turunan dari InputStream maka class FileInputStream menurunkan beberapa metode darinya. Seperti read(), skip(), dan close(). Class FileInputStream memiliki metode getFD() yang dipakai untuk mengambil object FileDescriptor.

Class FileOutputStream memungkinkan file ditulis sebagai output dalam bentuk stream. Sebagai turunan dari OutputStream maka class FileOutputStream

menurunkan beberapa metode darinya. Seperti write() dan close(). Class FileOutputStream memiliki metode getFD() yang dipakai untuk mengambil object FileDescriptor.

D. Tugas

Tugas 1

Buatlah suatu program sederhana untuk mengecek apakah suatu file sistem masih termuat pada suatu file directori!

❖ **Mengamati Listing Program dan Output Program**

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Jelaskan menegenai FileInputStream dan FileOutputStream!
2. Apa hubungannya class file dan FileDescriptor dengan FileInputStream atau FileOutputStream?
3. Terletak di direktori manakah file java2.txt setelah menjalankan program di bawah ini?

Listing Program

```
1  import java.io.FileOutputStream;
2  public class modul {
3  public static void main(String[] args) {
4  try{
5  FileOutputStream                    fos=new
   FileOutputStream("java2.txt");
6  String teks="saya belajar FileOutputStream";
7  byte[]isi=teks.getBytes();
8  fos.write(isi);
9  fos.close();
10 }
11 catch(Exception e){
12 System.err.println("Error tulis ke file");
13 }
14 }
15 }
```

F. Lembar Jawaban Tes Formatif

LJ- 01 : Pengertian FileInputStream dan FileOutputStream

a) FileInputStream

....

....

....

....

b) FileOutputStream

....

....

....

....

G. Lembar Kerja Siswa

18. Kegiatan 18 : Operasi File (Stream, Reader dan Writer)

i. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 18 ini siswa diharapkan dapat :

1. Memahami tentang stream, reader, dan writer
2. Menerapkannya pada suatu program

ii. Uraian Materi

1) Stream, Reader, dan Writer

Tanpa bisa berinteraksi dengan dunia lain, suatu program tidak ada gunanya. Interaksi suatu program dengan dunia lain sering disebut input/output atau I/O. Sejak dulu, salah satu tantangan terbesar untuk mendesain bahasa pemrograman baru adalah mempersiapkan fasilitas untuk melakukan input dan output. Komputer bisa terhubung dengan beragam jenis input dan output dari berbagai perangkat. Jika bahasa pemrograman harus dibuat secara khusus untuk setiap jenis perangkat, maka kompleksitasnya akan tak lagi bisa ditangani.
Salah satu kemajuan terbesar dalam sejarah pemrograman adalah adanya konsep (atau abstraksi) untuk memodelkan perangkat I/O. Dalam Java, abstraksi ini disebut dengan aliran (stream). Bagian ini akan memperkenalkan tentang aliran, akan tetapi tidak menjelaskan dengan komplit. Untuk lebih lengkapnya, silakan lihat dokumen resmi Java.

Ketika berhubungan dengan input/output, kita harus ingat bahwa ada dua kategori data secara umum : data yang dibuat oleh mesin, dan data yang bisa dibaca manusia. Data yang dibuat mesin ditulis dengan model yang sama dengan bagaimana data tersebut disimpan di dalam komputer, yaitu rangkaian nol dan satu. Data yang bisa dibaca manusia adalah data dalam bentuk rangkaian huruf. Ketika kita membaca suatu bilangan 3.13159, kita membacanya sebagai rangkaian huruf yang kita terjemahkan sebagai angka. Angka ini akan ditulis dalam komputer sebagai rangkaian bit yang kita tidak mengerti.

Untuk menghadapi kedua jenis data ini, Java memiliki dua kategori besar untuk aliran : aliran byte untuk data mesin (byte stream), dan aliran karakter (character

stream) untuk data yang bisa dibaca manusia. Ada banyak kelas yang diturunkan dari kedua kategori ini.Setiap objek yang mengeluarkandata ke aliran byte masuk sebagai kelas turunan dari kelas abstrak OutputStream. Objek yang membaca data dari aliran byte diturunkan dari kelas abstrak InputStream. Jika kita menulis angka ke suatu OutputStream, kita tidak akan bisa membaca data tersebut karena ditulis dalam bahasa mesin. Akan tetapi data tersebut bisa dibaca kembali oleh InputStream. Proses baca tulis data akan menjadi sangat efisien, karena tidak ada penerjemahan yang harus dilakukan : bit yang digunakan untuk menyimpan data di dalam memori komputer hanya dikopi dari dan ke aliran tersebut.

Untuk membaca dan menulis data karakter yang bisa dimengerti manusia, kelas utamanya adalah Reader dan Writer. Semua kelas aliran karakter merupakan kelas turunan dari salah satu dari kelas abstrak ini. Jika suatu angka akan ditulis dalam aliran Writer, komputer harus bisa menerjemahkannya ke dalam rangkaian karakter yang bisa dibaca maunsia.

Membaca angka dari aliran Reader menjadi variabel numerik juga harus diterjemahkan, dari deretan karakter menjadi rangkaian bit yang dimengerti komputer. (Meskipun untuk data yang terdiri dari karakter, seperti dari editor teks, masih akan ada beberapa terjemahan yang dilakukan. Karakter disimpan dalam komputer dalam nilai Unicode 16-bit. Bagi orang yang menggunakan alfabet biasa, data karakter biasanya disimpan dalam file dalam kode ASCII, yang hanya menggunakan 8-bit. Kelas Reader dan Writer akan menangani perubahan dari 16-bit ke 8-bit dan sebaliknya, dan juga menangani alfabet lain yang digunakan negara lain.)

Adalah hal yang mudah untuk menentukan apakah kita harus menggunakan aliran byte atau aliran karakter. Jika kita ingin data yang kita baca/tulis untuk bisa dibaca manusia, maka kita gunakan aliran karakter. Jika tidak, gunakan aliran byte. System.in dan System.out sebenarnya adalah aliran byte dan bukan aliran karakter, karenanya bisa menangani input selain alfabet, misalnya tombol enter, tanda panah, escape, dsb.

Kelas aliran standar yang akan dibahas berikutnya didefinisikan dalam paket java.io beserta beberapa kelas bantu lainnya. Kita harus mengimpor kelas-kelas tersebut dari paket ini jika kita ingin menggunakannya dalam program kita. Artinya dengan menggunakan "import java.io.*" di awal kode sumber kita.
Aliran tidak digunakan dalam GUI, karena GUI memiliki aliran I/O tersendiri. Akan tetapi kelas-kelas ini digunakan juga untuk file atau komunikasi dalam jaringan. Atau bisa juga digunakan untuk komunikasi antar thread yang sedang bekerja secara bersamaan. Dan juga ada kelas aliran yang digunakan untuk membaca dan menulis data dari dan ke memori komputer.

Class dasar I/O Reader, Writer, InputStream dan OutputStream hanya menyediakan operasi I/O sangat dasar. Misalnya, class InputStream memiliki metode instansi.

```
publicint read() throws IOException
```

Untuk membaca satu byte data dari aliran input. Jika sampai pada akhir dari aliran input , metode read() akan mengembalikan nilai -1. Jika ada kesalahan yang terjadi pada saat pengambilan input, maka pengecualian IOException akan dilemparkan. Karena IOException adalah class pengecualian yang harus ditangani, artinya kita harus menggunakan metode read() di dalam penyataan **try** atau mengeset subrutin untuk **throws** IOException.

Class InputStream juga memiliki metode untuk membaca beberapa byte data dalam satu langkah ke dalam array **byte**. Akan tetapi InputStream tidak memiliki metode untuk membaca jenis data lain, seperti **int** atau **double** dari aliran. Ini bukan masalah karena dalam prakteknya kita tidak akan menggunakan objek bertipe InputStream secara langsung. Yang akan kita gunakan adalah class turunan dari InputStream yang memiliki beberapa metode input yang lebih beragam daripada InputStream itu sendiri.

Begitu juga dengan class OutputStream memiliki metode output primitif untuk menulis satu byte data ke aliran output, yaitu metode

```
publicvoid write(int b) throws IOException
```

Tapi kita hampir pasti akan menggunakan class turunannya yang mampu menangani operasi yang lebih kompleks.Class Reader dan Writer memiliki operasi dasar yang hampir sama, yaitu read dan write, akan tetapi class ini berorientasi karakter (karena digunakan untuk membaca dan menulis data yang bisa dibaca manusia). Artinya operasi baca tulis akan mengambil dan menulis nilai **char** bukan byte. Dalam prakteknya kita akan menggunakan class turunan dari class-class dasar ini.

Salah satu hal menarik dari paket I/O pada Java adalah kemungkinan untuk menambah kompleksitas suatu aliran dengan membungkus aliran tersebut dalam objek aliran lain. Objek pembungkus ini juga berupa aliran, sehingga kita juga bisa melakukan baca tulis dari objek yang sama dengan tambahan kemampuan dalam objek pembungkusnya.Misalnya, PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusial. Jika kita memiliki objek bertipe Writer atau turunannya, dan kita ingin menggunakan metode pada PrintWriter untuk menulis data, maka kita bisa membungkus objek Writer dalam objek PrintWriter.

Contoh jika baskomKarakter bertipe Writer, maka kita bisa membuat

```
PrintWriter printableBaskomKarakter = new PrintWriter(baskomKarakter);
```

Ketika kita menulis data ke printableBaskomKarakter dengan menggunakan metode pada PrintWriter yang lebih canggih, maka data tersebut akan ditempatkan di tempat yang sama dengan apabila kita menulis langsung pada baskom Karakter. Artinya kita hanya perlu membuat antar muka yang lebih baik untuk aliran output yang sama. Atau dengan kata lain misalnya kita bisa menggunakan PrintWriter untuk menulis file atau mengirim data pada jaringan.

Untuk lengkapnya, metode pada class PrintWriter memiliki metode sebagai berikut :

Listing Program

```
1    // Metode untuk menulis data dalam
2    // bentuk yang bisa dibaca manusia
3    Public void print(String s)
4    public void print(char c)
5    public void print(int i)
6    public void print(long l)
7    public void print(float f)
8    public void print(double d)
9    public void print(boolean b)

10   // Menulis baris baru ke aliran
11   Public void println()
12   // Metode ini sama dengan di atas
13   // akan tetapi keluarannya selalu
14   // ditambah dengan baris baru
15   Public void println(String s)
16   Public void println(char c)
17   Public void println(int i)
18   Public void println(long l)
19   Public void println(float f)
20   Public void println(double d
21   Public void println(boolean b)
```

Catatan bahwa metode-metode di atas tidak pernah melempar pengecualian IOException. Akan tetapi, class PrintWriter memiliki metode

```
publicboolean checkError()
```

yang akan mengembalikan true jika ada kesalahan yang terjadi ketika menulis ke dalam aliran. Class PrintWriter menangkap pengecualian IOException secara internal, dan mengeset nilai tertentu di dalam class ini jika kesalahan telah terjadi. Sehingga kita bisa menggunakan metode pada PrintWriter tanpa khawatir harus menangkap pengecualian yang mungkin terjadi. Akan tetapi, jika kita ingin

membuat progam yang tangguh tentunya kita harus selalu memanggil checkError() untuk melihat apakah kesalahan telah terjadi ketika kita menggunakan salah satu metode pada PrintWriter.

Ketika kita menggunakan metode PrintWriter untuk menulis data ke aliran, data tersebut diubah menjadi rangkaian karakter yang bisa dibaca oleh manusia. Bagaimana caranya jika kita ingin membuat data dalam bentuk bahasa mesin?
Paket java.Io memiliki class aliran byte, yaitu DataOutputStream yang bisa digunakan untuk menulis suatu data ke dalam aliran dalam format biner. DataOutputStream berhubungan erat dengan OutputStream seperti hubungan antara PrintWriter dan Writer.

Artinya, OutputStream hanya berisi metode dasar untuk menulis byte, sedangkan DataOutputStream memiliki metode writeDouble(**double** x) untuk menulis nilai double, writeInt(**int** x) untuk menulis nilai int, dan seterusnya. Dan juga kita bisa membungkus objek bertipe OutputStream atau turunannya ke dalam aliran DataOutputStream sehingga kita bisa menggunakan metode yang lebih kompleks.
Misalnya, jika baskomByte adalah variabel bertipe OutputStream, maka

```
DataOutputStream baskomData = new DataOutputStream(baskomByte);
```

Untuk membungkus baskomByte dalam baskomData.Untuk mengambil data dari aliran, java.io memiliki class DataInputStream. Kita bisa membungkus objek bertipe InputStream atau turunannya ke dalam objek bertipe DataInputStream. Metode di dalam DataInputStream untuk membaca data biner bisa menggunakan readDouble(), readInt() dan seterusnya. Data yang ditulis oleh DataOutputStream dijamin untuk bisa dibaca kembali oleh DataInputStream, meskipun data kita tulis pada satu komputer dan data dibaca pada komputer jenis lain dengan sistem operasi berbeda. Kompatibilitas data biner pada Java adalah salah satu keunggulan Java untuk bisa dijalakan pada beragam platform.

Salah satu fakta yang menyedihkan tentang Java adalah ternyata Java tidak memiliki class untuk membaca data dalam bentuk yang bisa dibaca oleh

manusia. Dalam hal ini Java tidak memiliki class kebalikan dari PrintWriter sebagaimana DataOutputStream dan DataInputStream. Akan tetapi kita tetap bisa membuat class ini sendiri dan menggunakannya dengan cara yang persis sama dengan class-class di atas.

Class PrintWriter, DataInputStream, dan DataOutputStream memungkinkan kita untuk melakukan input dan output semua tipe data primitif pada Java. Pertanyaannya bagaimana kita melakukan baca tulis suatu objek?Mungkin secara tradisional kita akan membuat fungsi sendiri untuk memformat objek kita menjadi bentuk tertentu, misalnya urutan tipe primitif dalam bentuk biner atau karakter kemudian disimpan dalam file atau dikirim melalui jaringan. Proses ini disebut serialisasi (serializing) objek.
Pada inputnya, kita harus bisa membaca data yang diserialisasi ini sesuai dengan format yang digunakan pada saat objek ini diserialisasi. Untuk objek kecil, pekerjaan semacam ini mungkin bukan masalah besar. Akan tetapi untuk ukuran objek yang besar, hal ini tidak mudah.Akan tetapi Java memiliki cara untuk melakukan input dan output isi objek secaraotomatis,yaitu dengan menggunakan ObjectInputStream danObjectOutputStream.Class-class ini adalah class turunan dari InputStream danOutputStream yang bisa digunakan untuk membaca dan menulis objek yang sudah diserialisasi.ObjectInputStream dan ObjectOutputStream adalah class yang bisa dibungkus oleh class InputStream dan OutputStream lain. Artinya kita bisa melakukan input dan output objek pada aliran byte apa saja.Metde untuk objek I/O adalah readObject() yang tersedia pada ObjectInputStream dan writeObject(Object obj) yang tersedia dalam ObjectOutputStream. Keduanya bisa melemparkan IOException. Ingat bahwa readObject() mengembalikan nilai bertipe Object yang artinya harus di-type cast ke tipe sesungguhnya.ObjectInputStream dan ObjectOutputStream hanya bekerja untuk objek yang mengimplementasikan interface yang bernama Serializable. Lebih jauh semua variabel instansi pada objek harus bisa diserialisasi, karena interface Serializable tidak mempunyai metode apa-apa. Interface ini ada hanya sebagai penanda untuk kompiler supaya kompiler tahu bahwa objek ini digunakan untuk baca tulis ke suatu media.

Yang perlu kita lakukan adalah menambahkan "**implementsSerializable**" pada definisi class. Banyak class standar Java yang telah dideklarasikan untuk bisa diserialisasi, termasuk semua komponen class Swing dan AWT. Artinya komponen GUI pun bisa disimpan dan dibaca dari dalam perangkat I/O menggunakan ObjectInputStream dan ObjectOutputStream.

iii. Rangkuman

Interaksi suatu program dengan dunia lain sering disebut input/output atau I/O.
Salah satu kemajuan terbesar dalam sejarah pemrograman adalah adanya konsep (atau abstraksi) untuk memodelkan perangkat I/O. Dalam Java, abstraksi ini disebut dengan aliran (stream). Ada dua kategori data secara umum : data yang dibuat oleh mesin, dan data yang bisa dibaca manusia.
Java memiliki dua kategori besar untuk aliran : aliran byte untuk data mesin (byte stream), dan aliran karakter (character stream) untuk data yang bisa dibaca manusia. Untuk membaca dan menulis data karakter yang bisa dimengerti manusia, kelas utamanya adalah Reader dan Writer. Jika suatu angka akan ditulis dalam aliran Writer, komputer harus bisa menerjemahkannya ke dalam rangkaian karakter yang bisa dibaca manusia.
Membaca angka dari aliran Reader menjadi variabel numerik juga harus diterjemahkan, dari deretan karakter menjadi rangkaian bit yang dimengerti komputer.
Salah satu hal menarik dari paket I/O pada Java adalah kemungkinan untuk menambah kompleksitas suatu aliran dengan membungkus aliran tersebut dalam objek aliran lain.
PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusial.
class PrintWriter memiliki metode yang akan mengembalikan true jika ada kesalahan yang terjadi ketika menulis ke dalam aliran.
Salah satu fakta yang menyedihkan tentang Java adalah ternyata Java tidak memiliki class untuk membaca data dalam bentuk yang bisa dibaca oleh

manusia. Java tidak memiliki class kebalikan dari PrintWriter sebagaimana DataOutputStream dan DataInputStream.

ObjectOutputStream adalah class yang bisa dibungkus oleh class InputStream dan OutputStream lain. Komponen GUI bisa disimpan dan dibaca dari dalam perangkat I/O menggunakan ObjectInputStream dan ObjectOutputStream.

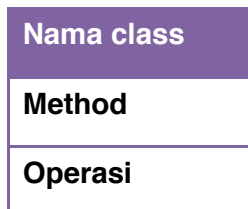
iv. Tugas

Tugas 1

Buatlah program menggunakan ObjectInputStream dan ObjectOutputStream untuk menampilkan data siswa yang terdiri dari :

- Nama
- NIS
- Kelas
- Program Studi

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

v. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang kamu ketahui tentang Stream, reader, dan writer?
2. Apa fungsi dari implementsSerializable?

vi. Lembar Jawaban Tes Formatif

LJ- 01 : Pengertian Stream, reader, dan writer

a) Stream

....

....

....

....

b) Reader

....

....

....

....

b) writer

....

....

....

....

LJ- 02 : Fungsi implementsSerializable

....

....

....

DAFTAR PUSTAKA

Christian Musnter, Java 2 JDK 5 –Grundlagen , Herdt – Verlag forbildungsmedien Gagh, Bodenheilm, 2006

C. Thomas wu, An Introduction to Object- Oriented Programming with Java, McGraw Hill 2001

Deitel, Java : How to program, Prentice Hall, New jersey, 2002

Joyce Avestro , Introduction Programming 1, Java Education Development Initiatif, 2003

Joyce Avestro , Introduction Programming 2, Java Education Development Initiatif, 2003

Patric Noughton , The Java Handbook, McGrawHill, Inc, 2006

R.H.Sianipar , Teori dan Implementasi Java, Informatika Bandung,2013